Support Vector Machine Classifier via $L_{0/1}$ Soft-Margin Loss

Huajun Wang, Yuanhai Shao, Shenglong Zhou, Ce Zhang and Naihua Xiu*

Abstract—Support vector machines (SVM) have drawn wide attention for the last two decades due to its extensive applications, so a vast body of work has developed optimization algorithms to solve SVM with various soft-margin losses. To distinguish all, in this paper, we aim at solving an ideal soft-margin loss SVM: $L_{0/1}$ soft-margin loss SVM (dubbed as $L_{0/1}$-SVM). Many of the existing (non)convex soft-margin losses can be viewed as one of the surrogates of the $L_{0/1}$ soft-margin loss. Despite its discrete nature, we manage to establish the optimality theory for the $L_{0/1}$-SVM including the existence of the optimal solutions, the relationship between them and P-stationary points. These not only enable us to deliver a rigorous definition of $L_{0/1}$ support vectors but also allow us to define a working set. Integrating such a working set, a fast alternating direction method of multipliers is then proposed with its limit point being a locally optimal solution to the $L_{0/1}$-SVM. Finally, numerical experiments demonstrate that our proposed method outperforms some leading classification solvers from SVM communities, in terms of faster computational speed and a fewer number of support vectors. The bigger the data size is, the more evident its advantage appears.

Index Terms—$L_{0/1}$ soft-margin loss, $L_{0/1}$-SVM, $L_{0/1}$ proximal operator, minimizer and P-stationary point, $L_{0/1}$ support vectors, $\mathrm{L}_{0/1}$ADMM.

1 Introduction

SUPPORT vector machines (SVM) were first introduced by Vapnik and Cortes [1] and then have been extensively applied into machine learning, statistic, pattern recognition and so forth. The basic idea of SVM is to find a maximum margin-type hyperplane in the input space that separates the training dataset. In the paper, we focus on the binary classification problem described as follows. Suppose we are given a training set $\{(\mathbf{x}_i, y_i) : i = 1, 2, \cdots, m\}$, where $\mathbf{x}_i \in \mathbb{R}^n$ are the input vectors and $y_i \in \{-1, 1\}$ are the output labels. The purpose of SVM is to train a hyperplane $\langle \mathbf{w}, \mathbf{x}\rangle + b = w_1x_1 + \cdots + w_nx_n + b = 0$ with $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ to be estimated by the training set. For any new input vector $\mathbf{x}'$, one can predict its label y' by $y' = 1$ if $\langle \mathbf{w}, \mathbf{x}'\rangle + b > 0$ and $y' = -1$ otherwise. In order to find an optimal hyperplane, there are two possible scenarios: linearly separable and inseparable training data. If the training data is linearly separated in the input space, then the unique optimal hyperplane can be obtained by solving a convex quadratic programming:

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1, i \in \mathbb{N}_m, \end{aligned} \tag{1}$$

where $\mathbb{N}_m := \{1, 2, \cdots, m\}$. The above model is known as the hard-margin SVM because it requires correct classifications of all training samples. When it comes to the training data being linearly inseparable in the input space, the popular approach is to allow violations in the satisfaction of the constraints in (1) and penalize such violations in the objective function, namely,

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \ell(1 - y_i f(\mathbf{x}_i)), \tag{2}$$

where $C > 0$ is a penalty parameter and $f(\mathbf{x}_i) := \langle \mathbf{w}, \mathbf{x}_i\rangle + b$. Here, $\ell(\cdot)$ is one of loss functions that aims at penalizing some sufficiently incorrectly classified samples and leaving the others. The above model is known as soft-margin SVM, allowing misclassified training samples. Authors in [1]–[3] have pointed out that the ideal soft-margin SVM is

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \ell_{0/1}(1 - y_i f(\mathbf{x}_i)), \tag{3}$$

where the soft-margin loss function $\ell_{0/1}(\cdot)$ is given by

$$\ell_{0/1}(t_i) = \begin{cases} 1, & 1 - t_i > 0, \\ 0, & 1 - t_i \leq 0, \end{cases} \tag{4}$$

and $t_i = y_i f(\mathbf{x}_i), i \in \mathbb{N}_m$. We name (3) as $L_{0/1}$-SVM, which minimizes the number of soft-margin misclassified samples. It is worth mentioning that the $\ell_{0/1}(\cdot)$ loss function arises in binary-valued regression, and is useful in many machine learning problems: candidates include those from perceptron learning [4], deep learning [5] and distributionally robust supervised learning [6]. However, the $L_{0/1}$-SVM is NP-hard [7], [8] since the $\ell_{0/1}(\cdot)$ loss is nonconvex and discontinuous, and up to now, it has not been fundamentally well investigated.

As far as we know, this is the first paper that establishes the optimality theory for the $L_{0/1}$-SVM and develops an

- *H.J. Wang, C. Zhang, N.H. Xiu are with the Department of Applied Mathematics, Beijing Jiaotong University, Beijing, P.R. China. Email: huajunwang@bjtu.edu.cn, czhang@bjtu.edu.cn, nhxiu@bjtu.edu.cn.*
- *Y.H. Shao is with the School of Management, Hainan University, Haikou, P.R. China. Email: shaoyuanhai@hainanu.edu.cn.*
- *S.L. Zhou is with the Department of Electrical and Electronic Engineering, Imperial College London, London, UK. Email: slzhou2021@163.com.*
- ** Corresponding author*

Manuscript received xx, xx; revised xx, xx.

effective algorithm aiming at pursuing an optimal solution to (3). The main contributions are summarized as follows.

(C1) We prove that the globally optimal solutions to the $L_{0/1}$-SVM exist and also establish its optimality condition aiming at finding such solutions. The condition has a close relationship to the P-stationary point which is very practical to solve the $L_{0/1}$-SVM, even though the problem is NP-hard.

(C2) Recall that the vector $\mathbf{w}^*$ that maximizes the margin can be shown to have the form:

$$\mathbf{w}^* = \alpha_1^* y_1 \mathbf{x}_1 + \cdots + \alpha_m^* y_m \mathbf{x}_m = \sum_{i:\ \alpha_i^* \neq 0} \alpha_i^* y_i \mathbf{x}_i, \qquad (5)$$

where $\boldsymbol{\alpha}^* = (\boldsymbol{\alpha}_1^*, \boldsymbol{\alpha}_2^*, ..., \boldsymbol{\alpha}_m^*)^\top$ is a solution to the dual problem of (1). The training vectors $\mathbf{x}_i$ corresponding to non-zero α_i^* are called support vectors [1], [9]. In this paper, the P-stationary point allows us to define the $L_{0/1}$ support vectors which coincide with the non-zero elements of the Lagrangian multiplier of (3). From the point of the optimization, the Lagrangian multiplier can be treated as a solution to the dual problem of (3), even though the dual problem is difficult to be derived due to the discreteness of $\ell_{0/1}(\cdot)$. Therefore, $L_{0/1}$ support vectors are standard support vectors. Furthermore, we show that all $L_{0/1}$ support vectors fall into the support hyperplanes $\langle \mathbf{w}^*, \mathbf{x} \rangle + b^* = \pm 1$, where $(\mathbf{w}^*, b^*)$ is a P-stationary point of (3). Hence, the number of $L_{0/1}$ support vectors are naturally expected to be no greater than the number of the standard support vectors. This is also testified by our numerical experiments.

(C3) When it comes to solving the problem (3), we adopt the famous alternating direction method of multipliers (ADMM), where one of its sub-problems is addressed by the $L_{0/1}$ proximal operator involved in the P-stationary point, which together with the idea of $L_{0/1}$ support vectors allows us to define a working set in each step. Indices i of vectors $\mathbf{x}_i$ out of this working set will be discarded, so the proposed method has a considerably low computational complexity and thus runs super fast. We prove that the limit point of the generated sequence is a P-stationary point and also a locally optimal solution to the problem (3). This means the final classifier only uses a small number of support vectors based on the statements in C2.

(C4) Comparing with some leading classification solvers for addressing the SVM problems on synthetic and real datasets, extensive numerical experiments demonstrate that our proposed method achieves better performance including higher prediction accuracy, a fewer number of support vectors and faster computational speed. In addition, the numerical comparison also certifies the robustness to the outliers of the $L_{0/1}$-SVM.

The remainder of this paper is organized as follows. In the next section, a brief overview of various soft-margin loss functions used in (2) will be given. Section 3 establishes the optimality theory including the existence of a globally optimal solution to the problem (3) and the relationships between a P-stationary point and an optimal solution. In Section 4, we will introduce the $L_{0/1}$ support vectors and cast a fast ADMM whose each step is integrated by a working set strategy inspired by the $L_{0/1}$ support vectors. Numerical experiments and concluding remarks are given in the last two sections.

2 Related work

The discrete nature of $\ell_{0/1}(\cdot)$ in $L_{0/1}$-SVM (3) limits its wide applications. Therefore, most previous work [10], [11] focus on the continuous surrogates of (3), namely, $\ell(\cdot)$ in (2) is a continuous approximation of $\ell_{0/1}(\cdot)$. We mention two typical classes of such surrogate soft-margin loss functions [3]. The first one consists of the convex soft-margin loss functions. An impressive body of work has designed such kinds of functions since they make the corresponding SVM problems easier to deal with. Here, we only review some popular ones.

- *Hinge soft-margin loss function:* $\ell_{\rm hinge}(t) = \max\{0, 1-t\}$. It is non-differentiable at $t = 1$ and unbounded. SVM with hinge soft-margin loss function was first proposed by Vapnik and Cortes [1], aiming at only penalizing the samples with $t < 1$. Hinge soft-margin loss SVM is the first SVM model and is widely studied by researchers [12].
- *Pinball soft-margin loss function:* $\ell^\tau_{\rm pinball}(t) = \max\{1-t, -\tau(1-t)\}$, with $0 \leq \tau \leq 1$, which is still non-differentiable at $t = 1$ and unbounded. SVM with this soft-margin loss function was proposed in [13], [14] to pay penalty for all training samples. There is a quadratic programming solver embedded in Matlab to solve the SVM with pinball soft-margin loss function [14].
- *Huberized hinge soft-margin loss function:* $\ell^\tau_{\rm HH}(t) = \max\{1-t-\tau/2, \min\{\max\{1-t, 0\}^2/2\tau, \tau/2\}\}$ with $\tau > 0$. It is smooth but still unbounded function. SVM with such soft-margin loss function was first proposed in [15] which can be solved by proximal gradient method [16].
- *Square soft-margin loss function [17], [18]:* $\ell_{\rm square}(t) = (1-t)^2$, a smooth but unbounded function.
- *Other convex and smooth soft-margin loss functions* include the squared hinge soft-margin loss function [19] and log soft-margin loss function [20].
- *Other convex and nonsmooth soft-margin loss functions* include the ε-insensitive zone pinball soft-margin loss function [14] and ϕ-risk hinge soft-margin loss function [21].

As the above loss functions are convex, their corresponding SVM models are not difficult to be dealt with [12]–[24]. However, the convexity often induces the unboundedness [25], [26], which weakens the robustness of those loss functions to outliers from the training data. To overcome such a drawback, one can set an upper bound and enforce the loss to stop increasing after a certain point. This gives rise to the second group: the nonconvex soft-margin loss functions.

- *Ramp soft-margin loss function [27], [28]:* $\ell^\mu_{\rm ramp}(t) = \max\{0, 1-t\} - \max\{0, 1-(t+\mu)\}$ with $\mu > 0$, which is non-differentiable at $t = 1-\mu$ and $t = 1$ but bounded between 0 and μ. It does not penalize the case when $t > 1$, while pays linear penalty when $1-\mu \leq t \leq 1$ and a fixed penalty μ when $t < 1-\mu$. This makes such a function robust to outliers.
- *Truncated pinball soft-margin loss function [29]* (truncated right side of pinball loss function): $\ell^{\tau,\kappa}_{\rm Tpin}(t) = \max\{0, (1+\tau)(1-t)\} - (\max\{0, \tau(1-t+\kappa)\} - \tau\kappa)$,

with $0 \leq \tau \leq 1$ and $\kappa \geq 0$. It is non-differentiable at $t = 1$ and $t = 1 + \kappa$ and unbounded. The penalty is fixed at κ for $t > 1 + \kappa$ and is linear otherwise.

- *Asymmetrical truncated pinball soft-margin loss function [30]* (truncated two side of pinball loss function): $\ell_{\mathrm{ATpin}}^{\tau,\kappa,\mu}(t) = \max\{0,(1+\tau)(1-t)\} - (\max\{0,\tau(1-t+\kappa)\} + \max\{0,1-t-\mu\} - \tau\kappa)$ with $0 \leq \tau \leq 1$ and $\mu, \kappa \geq 0$. This function is non-differentiable at $t = 1-\mu, t = 1+\kappa$ and $t = 1$ but bounded. The penalty is fixed at $\tau\kappa$ for $t > 1+\kappa$ and at μ for $t < 1-\mu$ but is linear otherwise.
- *Sigmoid soft-margin loss function [31]*: $\ell_{\mathrm{sigmoid}}(t) = 1/(1+\exp(-\tau(1-t))$ with $\tau > 0$. It is a smooth and bounded function. It penalizes all training samples.
- *Other nonconvex and smooth soft-margin loss functions* include the smooth ramp soft-margin loss function [32], savage loss [10] and One-sided cauchy soft-margin loss function [3], [33].
- *Other nonconvex and nonsmooth soft-margin loss functions* include the truncated logistic soft-margin loss function [34], curriculum loss [11] and ε-insensitive truncated least square soft-margin loss function [35].

Compared to convex soft-margin loss functions, most nonconvex ones are less sensitive to feature noise or outliers due to their boundedness. Apparently, nonconvexity would lead to difficulties of computations in terms of solving the corresponding SVM models [10], [11], [25]–[37].

3 Optimality Theory of $L_{0/1}$-SVM

For convenience of our subsequent analysis, denote

$$\begin{array}{rcl} A & := & [y_1\mathbf{x}_1\ y_2\mathbf{x}_2\ \cdots\ y_m\mathbf{x}_m]^\top \in \mathbb{R}^{m\times n}, \\ \mathbf{y} & := & (y_1, y_2, \cdots, y_m)^\top \in \mathbb{R}^m, \\ \mathbf{1} & := & (1, 1, \cdots, 1)^\top \in \mathbb{R}^m, \\ \mathbf{u} & := & \mathbf{1} - A\mathbf{w} - b\mathbf{y} \in \mathbb{R}^m, \\ \mathbf{u}_+ & := & ((u_1)_+, \cdots, (u_m)_+)^\top \in \mathbb{R}^m, \end{array} \tag{6}$$

where $t_+ := \max\{t, 0\}$. Moreover, the zero-norm of the vector $\mathbf{u}$ is denoted by $\|\mathbf{u}\|_0$ which counts the number of its non-zero elements. It is easy to see that $u_i = 1 - y_i\langle\mathbf{w}, \mathbf{x}_i\rangle - y_i b = 1 - y_i f(\mathbf{x}_i) = 1 - t_i, i \in \mathbb{N}_m$. Then the soft-margin loss function $\ell_{0/1}(\cdot)$ in (4) can be rewritten as

$$\ell_{0/1}(u_i) = \begin{cases} 1, & u_i > 0, \\ 0, & u_i \leq 0, \end{cases} \quad i \in \mathbb{N}_m. \tag{4'}$$

This indicates

$$\begin{array}{rcl} \displaystyle\sum_{i=1}^m \ell_{0/1}(1 - y_i f(\mathbf{x}_i)) & = & \displaystyle\sum_{i=1}^m \ell_{0/1}(u_i) \\ & = & \|\mathbf{u}_+\|_0 =: L_{0/1}(\mathbf{u}). \end{array} \tag{7}$$

Hence, the function $L_{0/1}(\mathbf{u}) = \|\mathbf{u}_+\|_0$ computes the number of all positive elements in $\mathbf{u}$. We call it the $L_{0/1}$ soft-margin loss function. Borrowing these notation, the $L_{0/1}$-SVM (3) is equivalent to the following optimization problem,

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} f(\mathbf{w}; b) := \frac{1}{2}\|\mathbf{w}\|^2 + C\|(\mathbf{1} - A\mathbf{w} - b\mathbf{y})_+\|_0, \tag{8}$$

or the following problem with an extra variable $\mathbf{u}$,

$$\begin{array}{ll} \displaystyle\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}, \mathbf{u}\in\mathbb{R}^m} & \dfrac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0 \\ \text{s.t.} & \mathbf{u} + A\mathbf{w} + b\mathbf{y} = \mathbf{1}. \end{array} \tag{9}$$

Recall the sparse optimization problem $\min_{\mathbf{v}\in\mathbb{R}^m}\{g(\mathbf{v}) + C\|\mathbf{v}\|_0\}$, where $C > 0$ is a given penalty parameter and $g : \mathbb{R}^m \to \mathbb{R}$ is smooth or nonsmooth function. Due to the combinatorial nature of $\|\mathbf{v}\|_0$, the above sparse optimization problem is generally NP-hard. However, this problem has wide applications in linear and nonlinear compressive sensing, robust linear regression, deep learning, etc. Hence it has been extensively studied by a lot of researchers in different communities. More recently, by utilizing continuous optimization theory, the optimality conditions and algorithms for such a problem are successfully established by some researchers in optimization community [38]–[43].

Observe the $L_{0/1}$-SVM model (8) or (9). We found that it has same structure as the above sparse optimization model with difference between $\|(\cdot)_+\|_0$ and $\|(\cdot)\|_0$. Similarly, by utilizing continuous optimization theory, we do the optimality analysis of (8) or (9) in this section.

3.1 Existence of $L_{0/1}$-SVM Minimizer

Firstly, we show the existence of a global minimizer (a minimizer is often phrased as an optimal solution) to (8), a premise of the optimality condition of the $L_{0/1}$-SVM.

Theorem 3.1. Given $b \in \mathcal{I} := [-M, M]$ with $0 < M < +\infty$. Then the globally optimal solution to (8) exists and the solution set is bounded.

The proof of Theorem 3.1 is given in Supplement S.1. For any $b \in \mathcal{I}$, since $y_i \in \{-1, 1\}$, we have the following observations

$$f(\mathbf{0}; b) = C\|(\mathbf{1} - b\mathbf{y})_+\|_0 = \begin{cases} Cm_-, & b \geq 1, \\ Cm_+, & b \leq -1, \\ Cm, & |b| < 1, \end{cases}$$

where m_+ and m_- are the number of positive and negative y_i. Therefore, let $(\mathbf{w}^*; b^*)$ be an optimal solution to (8) (such a solution exists by Theorem 3.1), then

$$f(\mathbf{w}^*; b^*) \leq C\min\{m_+, m_-\}.$$

In numerical experiments, this gives us a clue to set some starting points $(\mathbf{w}^0; b^0)$ satisfying

$$f(\mathbf{w}^0; b^0) \leq C\min\{m_+, m_-\}. \tag{10}$$

3.2 First-Order Optimality Condition

From the perspective of optimization, establishing the optimality conditions of an optimization problem is a key step in theoretical analysis, because those conditions effectively benefits for the algorithmic design. Now turn our attention on the $L_{0/1}$-SVM model (9).

Definition 3.1 (P-stationary point of (9)). For a given $C > 0$, we say $(\mathbf{w}^*; b^*; \mathbf{u}^*)$ is a proximal stationary (P-stationary) point of (9) if there is a Lagrangian multiplier $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ and a constant $\gamma > 0$ such that

$$\begin{cases} \mathbf{w}^* + A^\top\boldsymbol{\lambda}^* = \mathbf{0}, \\ \langle\mathbf{y}, \boldsymbol{\lambda}^*\rangle = 0, \\ \mathbf{u}^* + A\mathbf{w}^* + b^*\mathbf{y} = \mathbf{1}, \\ \mathrm{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*) = \mathbf{u}^*, \end{cases} \tag{11}$$

where

$$[\text{Prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{z}^*)]_i = \begin{cases} 0, & 0 < z_i^* \leq \sqrt{2\gamma C}, \\ z_i^*, & z_i^* > \sqrt{2\gamma C} \text{ or } z_i^* \leq 0, \end{cases} \quad (12)$$

and $\mathbf{z}^* := \mathbf{u}^* - \gamma\boldsymbol{\lambda}^*$. The above equation (12) is termed as $L_{0/1}$ proximal operator, whose solution has been derived in Supplement S.2.

The $L_{0/1}$ proximal operator is the key in the optimality analysis (see Theorem 3.2 below) and algorithmic design (see Section 4.2) of $L_{0/1}$-SVM. Using the above definition, we reveal the relationship between local/global minimizer and a P-stationary point of $L_{0/1}$-SVM. To proceed more, let

$$B := [A\ \mathbf{y}] \in \mathbb{R}^{m\times(n+1)}, \quad H := \begin{bmatrix} I_{n\times n} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} B^+, \quad (13)$$

where $B^+ \in \mathbb{R}^{(n+1)\times m}$ is the generalized inverse of B, and $\lambda_H := \lambda_{\max}(H^\top H)$ where $\lambda_{\max}(H^\top H)$ is the maximum eigenvalue of $H^\top H$. Thus, we have following theorem.

Theorem 3.2. The following relations hold for (9).

(i) A globally optimal solution is also a P-stationary point with $0 < \gamma < 1/\lambda_H$ if B is full column rank.
(ii) A P-stationary point with $\gamma > 0$ is also a locally optimal solution.

The proof of Theorem 3.2 is given in Supplement S.3. Note that B being full column rank implies $m > n$, i.e., the number of samples is greater than the number of features. However, from Theorem 3.2 (ii), if we find a P-stationary point of the problem (9), then it must be a locally optimal solution without any assumptions. No requirement of $m > n$ is enforced. Our numerical experiments testify that our proposed algorithm based on the idea of the P-stationary point works well for both cases: $m > n$ and $m \leq n$.

3.3 Extension

In Section 3.2, we established the first-order optimality condition for (9), i.e., (8), which is an unconstrained optimization problem. This can be regarded as a special case of the following general optimization model

$$\min_{\mathbf{u}\in\mathbb{R}^m} \quad g(\mathbf{u}) + C\|\mathbf{u}_+\|_0, \quad (14)$$

where $C > 0$ is a given penalty parameter and $g : \mathbb{R}^m \to \mathbb{R}$ is a smooth function and gradient Lipschitz continuous with a Lipschitz constant $\gamma_g > 0$.

Similarly, we introduce the proximal stationary point of (14) as below.

Definition 3.2 (P-stationary point of (14)). For a given $C > 0$, we say $\mathbf{u}^*$ is a proximal stationary (P-stationary) point of problem (14) if there is a constant $\gamma > 0$ such that

$$\mathbf{u}^* \quad = \text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\nabla g(\mathbf{u}^*)), \quad (15)$$

where, $\nabla g(\cdot)$ is the gradient of $g(\cdot)$.

The following theorem reveals the relationship between a local/global minimizer and a P-stationary point of (14), whose the proof is similar to that of the Theorem 3.2 and thus is omitted.

Theorem 3.3. For problem (14), the following relations hold.

(i) For a given $C > 0$, if $\mathbf{u}^*$ is a global minimizer of (14) then it is a P-stationary point with $0 < \gamma < 1/\gamma_g$.
(ii) For a given $C > 0$, if g is convex and $\mathbf{u}^*$ is a P-stationary point with $\gamma > 0$, then it is a local minimizer of (14).

The above two theorems state that under condition of convexity, the P-stationary point must be a local minimizer, which means that we could use the P-stationary point as a termination rule in terms of guaranteeing the local optimality of a point generated by the algorithm proposed in next section.

4 Fast Algorithm

It is well known that the classifier is decided by support vectors, see (5). If support vectors is used to design the solving algorithm, the fewer number of support vectors is, the faster the computational speed will be since fewer samples in training data are used to train the classifier. Therefore, reducing the number of support vectors tends to be important for datasets in extremely large sizes. Motivated by this, we introduce $L_{0/1}$ support vectors and working set strategy based on the theory in Section 3.2 and adopt the famous alternating direction method of multipliers (ADMM) to solve the $L_{0/1}$-SVM (9).

4.1 $L_{0/1}$ Support Vectors

Let $(\mathbf{w}^*; b^*; \mathbf{u}^*)$ be a P-stationary point of problem (9). Then from Definition 3.1, there is a Lagrangian multiplier $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ and a constant $\gamma > 0$ such that (11) holds. Let

$$T_* := \left\{ i \in \mathbb{N}_m : \ \mathbf{u}_i^* - \gamma\boldsymbol{\lambda}_i^* \in (0, \sqrt{2\gamma C}] \right\}, \quad (16)$$

and $\overline{T}_* := \mathbb{N}_m \backslash T_*$ be its complementarity set. Let $\mathbf{z}_T \in \mathbb{R}^{|T|}$ be the sub-vector of $\mathbf{z}$ indexed on T and $|T|$ be the cardinality of T. It follows from the last equation of (11) and (12) that

$$\begin{aligned} \mathbf{u}^* &\overset{(11)}{=} \text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*) \\ &= \begin{bmatrix} (\text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*))_{T_*} \\ (\text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*))_{\overline{T}_*} \end{bmatrix} \\ &\overset{(12)}{=} \begin{bmatrix} \mathbf{0}_{T_*} \\ (\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*)_{\overline{T}_*} \end{bmatrix}. \end{aligned}$$

which is equivalent to

$$\begin{bmatrix} \mathbf{u}_{T_*}^* \\ \boldsymbol{\lambda}_{\overline{T}_*}^* \end{bmatrix} = \mathbf{0}. \quad (17)$$

Then T_* in (16) turns to

$$T_* = \left\{ i \in \mathbb{N}_m : \ \lambda_i^* \in \Big[-\sqrt{2C/\gamma}, 0 \Big) \right\}. \quad (18)$$

This and (17) result in

$$\boldsymbol{\lambda}_i^* \begin{cases} \in [-\sqrt{2C/\gamma}, 0), & \text{for } i \in T_*, \\ = 0, & \text{for } i \in \overline{T}_*. \end{cases} \quad (19)$$

Taking (19) into the first equation of (11) derives

$$\begin{aligned} \mathbf{w}^* &= -A_{T_*}^\top \boldsymbol{\lambda}_{T_*}^* - A_{\overline{T}_*}^\top \boldsymbol{\lambda}_{\overline{T}_*}^* \\ &= -A_{T_*}^\top \boldsymbol{\lambda}_{T_*}^* = \sum_{i\in T_*} -\lambda_i^* y_i \mathbf{x}_i. \end{aligned} \quad (20)$$

Remark 4.1. Regarding the expression (20), we have the following comments.

- Recall (5), where $\boldsymbol{\alpha}^*$ is a solution to the dual problem of (1). From the optimization perspective, the Lagrangian multiplier $-\boldsymbol{\lambda}^*$ actually is a solution to the dual problem of (9). In such a sense, $\{\mathbf{x}_i : i \in T_*\}$ indeed are standard support vectors. While we call them the $L_{0/1}$ support vectors since they are selected by the $L_{0/1}$ proximal operator.
- Furthermore, the third equation in (11) implies $\mathbf{1} = \mathbf{u}^*_{T_*} + (A\mathbf{w}^* + b^*\mathbf{y})_{T_*} = (A\mathbf{w}^* + b^*\mathbf{y})_{T_*}$ due to $\mathbf{u}^*_{T_*} = 0$ by (17), which and the definition (6) of A yield

$$\langle \mathbf{w}^*, \mathbf{x}_i \rangle + b^* = \pm 1, \text{ for } i \in T_*. \tag{21}$$

  Interestingly, the $L_{0/1}$ support vectors must fall into the support hyperplanes $\langle \mathbf{w}^*, \mathbf{x} \rangle + b^* = \pm 1$. As far as we know, the hard-margin SVM has such a property for linearly separable datasets. For linearly inseparable datasets, most soft-margin SVM can not guarantee this property. However, (21) is ensured by the $L_{0/1}$-SVM regardless of the datasets being separable or inseparable. This phenomenon manifests that the $L_{0/1}$-SVM could render fewer support vectors than the other soft-margin SVM models, which is also certified by our numerical experiments.

The set T_* in (18) gives us a clue to select support vectors, which is very practical in the following algorithmic design.

4.2 $L_{0/1}$ADMM via Selection of Working Set

In this subsection, we take advantages of ADMM and working set to solve the $L_{0/1}$-SVM (9). We firstly give the framework of ADMM as follows. The augmented Lagrangian function of the problem (9) is given by

$$\begin{aligned} L_\sigma(\mathbf{w}; b; \mathbf{u}; \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0 + \langle \boldsymbol{\lambda}, \mathbf{u} - \mathbf{1} + A\mathbf{w} + b\mathbf{y} \rangle \\ + \frac{\sigma}{2}\|\mathbf{u} - \mathbf{1} + A\mathbf{w} + b\mathbf{y}\|^2, \end{aligned}$$

where $\boldsymbol{\lambda}$ is the Lagrangian multiplier and $\sigma > 0$ is the penalty parameter. Given the kth iteration $(\mathbf{w}^k; b^k; \mathbf{u}^k; \boldsymbol{\lambda}^k)$, the framework to update each component is as follows:

$$\begin{aligned} \mathbf{u}^{k+1} &= \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^m} L_\sigma(\mathbf{w}^k, b^k, \mathbf{u}, \boldsymbol{\lambda}^k) \\ \mathbf{w}^{k+1} &= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^n} L_\sigma(\mathbf{w}, b^k, \mathbf{u}^{k+1}, \boldsymbol{\lambda}^k) + \tfrac{\sigma}{2}\|\mathbf{w} - \mathbf{w}^k\|^2_{D_k} \\ b^{k+1} &= \operatorname*{argmin}_{b\in\mathbb{R}} L_\sigma(\mathbf{w}^{k+1}, b, \mathbf{u}^{k+1}, \boldsymbol{\lambda}^k) \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \eta\sigma(\mathbf{u}^{k+1} - \mathbf{1} + A\mathbf{w}^{k+1} + b^{k+1}\mathbf{y}), \end{aligned} \tag{22}$$

where $\eta > 0$ is the dual step-size. The proximal term is

$$\|\mathbf{w} - \mathbf{w}^k\|^2_{D_k} = \langle \mathbf{w} - \mathbf{w}^k, D_k(\mathbf{w} - \mathbf{w}^k) \rangle.$$

Note that if D_k is positive semidefinite, then the above framework is the standard semi-proximal ADMM [44]. However, authors in papers [45], [46] have also investigated ADMM with the indefinite proximal terms, namely, D_k is indefinite. The basic principle of choosing D_k is to guarantee the convexity of $\mathbf{w}$-subproblem of (22). Since $L_\sigma(\mathbf{w}; b^k; \mathbf{u}^{k+1}; \boldsymbol{\lambda}^k)$ is strongly convex with respect to $\mathbf{w}$, D_k is flexible to be chosen as an indefinite matrix.

Now, let's see how T_* in (18) instructs to select the support vectors. Denote $\mathbf{z}^k := \mathbf{1} - A\mathbf{w}^k - b^k\mathbf{y} - \boldsymbol{\lambda}^k/\sigma$. Define a working set T_k at the kth step by

$$T_k := \left\{ i \in \mathbb{N}_m : \ z_i^k \in \left(0, \sqrt{2C/\sigma}\ \right] \right\} \tag{23}$$

and $\overline{T}_k := \mathbb{N}_m \backslash T_k$. Based on which, D_k is chosen as

$$D_k = -A^\top_{\overline{T}_k} A_{\overline{T}_k}. \tag{24}$$

Here, for a given set $T \subseteq \mathbb{N}_m$, $A_T \in \mathbb{R}^{|T|\times n}$ denotes the submatrix containing rows of A indexed on T. The working set T_k and the choice of D_k will tremendously speed up the whole computation in each step of ADMM. More precisely, we calculate each sub-problem in (22) as follows.

(i) Updating $\mathbf{u}^{k+1}$: The $\mathbf{u}$-subproblem in (22) is equivalent to the following problem

$$\begin{aligned} & \mathbf{u}^{k+1} \\ = \ & \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^m} C\|\mathbf{u}_+\|_0 + \langle \boldsymbol{\lambda}^k, \mathbf{u} \rangle + \frac{\sigma}{2}\|\mathbf{u} - \mathbf{1} + A\mathbf{w}^k + b^k\mathbf{y}\|^2 \\ = \ & \operatorname*{argmin}_{\mathbf{u}\in\mathbb{R}^m} C\|\mathbf{u}_+\|_0 + \frac{\sigma}{2}\|\mathbf{u} - \mathbf{z}^k\|^2 \\ = \ & \mathrm{Prox}_{\frac{C}{\sigma}\|(\cdot)_+\|_0}(\mathbf{z}^k), \end{aligned}$$

where the last equation is from (12) with $\gamma = 1/\sigma$. This together with (12) and the working set (23) suffices to

$$\mathbf{u}^{k+1}_{T_k} = \mathbf{0}, \quad \mathbf{u}^{k+1}_{\overline{T}_k} = \mathbf{z}^k_{\overline{T}_k}. \tag{25}$$

Therefore, updating $\mathbf{u}^{k+1}$ turns to be very simple and fast.

(ii) Updating $\mathbf{w}^{k+1}$. The $\mathbf{w}$-subproblem in (22) is

$$\begin{aligned} \mathbf{w}^{k+1} = \arg\min_{\mathbf{w}\in\mathbb{R}^n} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{\sigma}{2}\|\mathbf{w} - \mathbf{w}^k\|^2_{-A^\top_{\overline{T}_k} A_{\overline{T}_k}} \\ + \langle \boldsymbol{\lambda}^k, A\mathbf{w} \rangle + \frac{\sigma}{2}\|\mathbf{u}^{k+1} - \mathbf{1} + A\mathbf{w} + b^k\mathbf{y}\|^2. \end{aligned} \tag{26}$$

It is a convex quadratic programming problem. To solve (26), we only need to find a solution to the equations

$$\begin{aligned} \mathbf{0} \ &= \ \mathbf{w} - \sigma A^\top_{\overline{T}_k} A_{\overline{T}_k}(\mathbf{w} - \mathbf{w}^k) + A^\top \boldsymbol{\lambda}^k \\ &+ \ \sigma A^\top(\mathbf{u}^{k+1} - \mathbf{1} + A\mathbf{w} + b^k\mathbf{y}), \end{aligned} \tag{27}$$

which is equivalent to find a solution to the equations

$$(I + \sigma A^\top_{T_k} A_{T_k})\mathbf{w} = \sigma A^\top_{T_k} \mathbf{v}^k_{T_k}, \tag{28}$$

where $\mathbf{v}^k := -(\mathbf{u}^{k+1} + b^k\mathbf{y} - \mathbf{1} + \boldsymbol{\lambda}^k/\sigma)$. To derive (28) from (27), we used two facts that $\mathbf{u}^{k+1}_{\overline{T}_k} = \mathbf{z}^k_{\overline{T}_k}$ by (25) and

$$A^\top_{T_k} A_{T_k} = A^\top A - A^\top_{\overline{T}_k} A_{\overline{T}_k}.$$

Therefore, the term $A_{\overline{T}_k}$ vanishes in (28), which means the working set T_k and the choice of D_k discard the samples $\{\mathbf{x}_j, j \in \overline{T}_k\}$. This would fasten the computation significantly if the selected $|T_k|$ is very small. In practice, (28) can be addressed efficiently by the following rules:

- If $n \le |T_k|$, one could solve (28) directly through

$$\mathbf{w}^{k+1} = (I + \sigma A^\top_{T_k} A_{T_k})^{-1} \sigma A^\top_{T_k} \mathbf{v}^k_{T_k}. \tag{29}$$

- If $n > |T_k|$, the Sherman-Morrison-Woodbury formula [47] enables us to calculate the inverse as

$$(I + \sigma A^\top_{T_k} A_{T_k})^{-1} = I - \sigma A^\top_{T_k}(I + \sigma A_{T_k} A^\top_{T_k})^{-1} A_{T_k}.$$

Then we update $\mathbf{w}^{k+1}$ by

$$\mathbf{w}^{k+1} = \sigma A_{T_k}^\top (I + \sigma A_{T_k} A_{T_k}^\top)^{-1} \mathbf{v}_{T_k}^k. \tag{30}$$

(iii) Updating b^{k+1}. The b-subproblem in (22) is a convex quadratic programming

$$b^{k+1} = \arg\min_{b\in\mathbb{R}} \ \langle \boldsymbol{\lambda}^k, b\mathbf{y}\rangle + \frac{\sigma}{2}\|\mathbf{u}^{k+1} - \mathbf{1} + A\mathbf{w}^{k+1} + b\mathbf{y}\|^2.$$

which is solved by

$$b^{k+1} = \langle \mathbf{y}, \mathbf{r}^k\rangle / \|\mathbf{y}\|^2 = \langle \mathbf{y}, \mathbf{r}^k\rangle / m, \tag{31}$$

where $\mathbf{r}^k := -A\mathbf{w}^{k+1} + \mathbf{1} - \mathbf{u}^{k+1} - \boldsymbol{\lambda}^k/\sigma$.

(iv) Updating $\boldsymbol{\lambda}^{k+1}$. We update $\boldsymbol{\lambda}^{k+1}$ in (22) as follows

$$\boldsymbol{\lambda}_{T_k}^{k+1} = \boldsymbol{\lambda}_{T_k}^k + \eta\sigma\boldsymbol{\varpi}_{T_k}^{k+1}, \quad \boldsymbol{\lambda}_{\overline{T}_k}^{k+1} = \mathbf{0}, \tag{32}$$

where $\boldsymbol{\varpi}^{k+1} := \mathbf{u}^{k+1} - \mathbf{1} + A\mathbf{w}^{k+1} + b^{k+1}\mathbf{y}$ and setting $\boldsymbol{\lambda}_{\overline{T}_k}^{k+1} = \mathbf{0}$ follows the idea in (17), namely, the part of the Lagrangian multiplier not on the working set is removed.

Overall, updating each subproblem is summarized into Algorithm 1, which is called $\mathtt{L_{0/1}ADMM}$, an abbreviation for $L_{0/1}$-SVM solved by ADMM.

Algorithm 1 : $\mathtt{L_{0/1}ADMM}$ for solving problem (9)

Initialize $(\mathbf{w}^0; b^0; \mathbf{u}^0; \boldsymbol{\lambda}^0)$. Set $C, \eta, \sigma, K > 0$ and $k = 0$.
while The halting condition does not hold and $k \le K$ **do**
  Update T_k as in (23).
  Update $\mathbf{u}^{k+1}$ by (25).
  Update $\mathbf{w}^{k+1}$ by (29) if $n \le |T_k|$ and by (30) otherwise.
  Update b^{k+1} by (31).
  Update $\boldsymbol{\lambda}^{k+1}$ by (32).
  Set $k = k + 1$.
end while
return the final solution $(\mathbf{w}^k, b^k)$ to (9).

4.3 Convergence and Complexity Analysis

The following theorem shows that if the sequence generated by $\mathtt{L_{0/1}ADMM}$ has a limit point, then it must be a P-stationary point and also a locally optimal solution to (9).

Theorem 4.1. Suppose $(\mathbf{w}^*; b^*; \mathbf{u}^*; \boldsymbol{\lambda}^*)$ be the limit point of the sequence $\{(\mathbf{w}^k; b^k; \mathbf{u}^k; \boldsymbol{\lambda}^k)\}$ generated by $\mathtt{L_{0/1}ADMM}$. Then $(\mathbf{w}^*; b^*; \mathbf{u}^*)$ is a P-stationary point with $\gamma = 1/\sigma$ and also a locally optimal solution to the problem (9).

The proof of Theorem 4.1 is given in Supplement S.4. Based on the authors' limited knowledge, the above convergence result is difficult to improve, because our $\mathtt{L_{0/1}ADMM}$ deals with $L_{0/1}$-SVM directly, whose objective function involves a discrete part $\|(\cdot)_+\|_0$. As a supplement, we mention some works on ADMM and its convergence analysis: for solving nonconvex nonsmooth optimization problems, see, e.g. [48]–[51]; for solving the nonconvex soft-margin loss SVMs, see, e.g. [52], [53].

With regard to the computational complexity in each iteration of the proposed algorithm $\mathtt{L_{0/1}ADMM}$, we have the following observations:

- Updating T_k by (23) needs the complexity $\mathcal{O}(m)$.
- The main term involved in computing $\mathbf{u}^{k+1}$ by (25) is $A\mathbf{w}^k$, taking the complexity about $\mathcal{O}(mn)$.
- To update $\mathbf{w}^{k+1}$, we compute (29) if $n \le |T_k|$ and (30) otherwise. For the former, the dominant computations are calculating

  $$A_{T_k}^\top A_{T_k} \ \text{ and } \ (I + \sigma A_{T_k}^\top A_{T_k})^{-1}.$$

  Their computational complexities are $\mathcal{O}(n^2|T_k|)$ and $\mathcal{O}(n^\kappa)$ with $\kappa \in (2, 3)$, respectively. For the latter, the dominant computations are from

  $$A_{T_k} A_{T_k}^\top \ \text{ and } \ (I + \sigma A_{T_k} A_{T_k}^\top)^{-1}$$

  with the computational complexities $\mathcal{O}(n|T_k|^2)$ and $\mathcal{O}(|T_k|^\kappa)$ with $\kappa \in (2, 3)$, respectively. Therefore, the complexity to update $\mathbf{w}^{k+1}$ in each step is

  $$\mathcal{O}(\min\{n^2, |T_k|^2\}\max\{n, |T_k|\}).$$

- Similarly, $A\mathbf{w}^{k+1}$ is the most expensive computation in (31) to derive b^{k+1}. Again its complexity is $\mathcal{O}(mn)$.
- Same as that of updating b^{k+1}, achieving $\boldsymbol{\lambda}^{k+1}$ by (32) takes $\mathcal{O}(mn)$ complexity.

Overall, the whole computational complexity in each step of $\mathtt{L_{0/1}ADMM}$ in Algorithm 1 is

$$\mathcal{O}\left(mn + \min\{n^2, |T_k|^2\}\max\{n, |T_k|\}\right).$$

If the selected working sets have low cardinalities $|T_k|$ or n is very small (i.e., $n \ll m$), $\mathtt{L_{0/1}ADMM}$ possesses a considerably low computational complexity.

With regard to non-asymptotic analysis for finding stationary points of nonsmooth nonconvex functions, see, e.g. [54].

5 Numerical Experiments

In this section, we conduct numerical experiments to show the sparsity, robustness and effectiveness of the proposed $\mathtt{L_{0/1}ADMM}$ (available at `https://github.com/Huajun-Wang/L01ADMM`) by using MATLAB (2018b) on a laptop of 32GB of memory and Inter Core i7 2.7Ghz CPU, against nine leading solvers on synthetic data and real data.

Inspired by Theorem 3.2, the P-stationary point is taken as a stopping criteria in the experiments. In the implementation, we terminate the proposed algorithm if the point $(\mathbf{w}^k; b^k; \mathbf{u}^k; \boldsymbol{\lambda}^k)$ closely satisfies the conditions in (11), i.e.,

$$\max\{\theta_1^k, \theta_2^k, \theta_3^k, \theta_4^k\} < \mathtt{tol},$$

where $\mathtt{tol}$ is the tolerance level and

$$\begin{aligned}
\theta_1^k &:= \frac{\|\mathbf{w}^k + A_{T_k}^\top \boldsymbol{\lambda}_{T_k}^k\|}{1 + \|\mathbf{w}^k\|}, \quad \theta_2^k := \frac{|\langle \mathbf{y}_{T_k}, \boldsymbol{\lambda}_{T_k}^k\rangle|}{1 + |T_k|}, \\
\theta_3^k &:= \frac{\|\mathbf{u}^k - \mathbf{1} + A\mathbf{w}^k + b^k\mathbf{y}\|}{\sqrt{m}}, \\
\theta_4^k &:= \frac{\|\mathbf{u}^k - \mathrm{prox}_{C/\sigma\|(\cdot)_+\|_0}(\mathbf{u}^k - \boldsymbol{\lambda}^k/\sigma)\|}{1 + \|\mathbf{u}^k\|}.
\end{aligned}$$

(a) Parameters setting. In our algorithm, the parameters C and σ control the number of support vectors (see (23)), so tuning good choices of these two parameters is crucial. Hence, the standard 10-fold cross validation is employed in training datasets to select them, where C is picked from $\{2^{-7}, 2^{-6}, \cdots, 2^7\}$ and σ is tuned from $\{a^{-7}, a^{-6}, \cdots, a^7\}$

with $a = \sqrt{2}$. The parameters with the highest cross validation accuracy are picked out. In addition, we set $\eta = 1.618$, maximum iteration number $K = 10^3$ and the tolerance level $\texttt{tol}= 10^{-3}$. For the starting points, set $\mathbf{u}^0 = \boldsymbol{\lambda}^0 = \mathbf{0}$. As mentioned in Section 3.1, we choose $\mathbf{w}^0 = \mathbf{1}/100$ and $b^0 = 0$ if it meets (10), and $\mathbf{w}^0 = 0$ and $b^0 = 1$ (or -1) otherwise.

(b) Benchmark classifiers. There is an impressive body of algorithms that have been developed to solve classification problems. However, to conduct fair comparisons, we only select nine solvers that were programmed by MATLAB. Eight of them address the SVM problem and one deals with the L_2-regularized logistic regression problem. All their parameters are also optimized by 10-fold cross validation to maximize accuracy.

- `HSVM` SVM with the hinge soft-margin loss is implemented by LibSVM ([55], `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`), where the parameter C is selected from $\Omega := \{2^{-7}, 2^{-6}, \cdots, 2^7\}$.
- `LSVM` SVM with the square soft-margin loss [17] is implemented by LibLSSVM ([56], `https://www.esat.kuleuven.be/sista/lssvmlab/`), where the parameter C is selected from Ω.
- `PSVM` SVM with the pinball soft-margin loss can be tackled by the traversal algorithm ([57], `https://www.esat.kuleuven.be/stadius/ADB/huang/softwarePINSVM.php`), where C is turned from a union of Ω and the one in [57] and τ is set as $\{-1, -0.99, \cdots, 0.99\}$ from [57].
- `RSVM` SVM with the ramp soft-margin loss can be addressed by `CCCP` ([27], `https://github.com/RampSVM/RSVM`), where the core subproblem of `CCCP` is solved by the MATLAB built-in function `quadprog`, while C and μ are selected from Ω and $\{0.1, 0.2, \cdots, 1\}$.
- `SSVM` SVM with the one-sided Cauchy soft-margin loss is solved by the iteratively reweighted algorithm (`IRA` [3], `https://www.esat.kuleuven.be/stadius/ADB/feng/softwareRSVC.php`). The key subproblem of `IRA` is solved by the CVX, and both C and ν are tuned from Ω.
- `LOGI` L_2-regularized logistic regression is addressed by employing Newton algorithm ([58], `https://github.com/tminka/logreg/`), where the parameter C is selected from Ω.
- `PEGA` SVM with the hinge soft-margin loss is solved by employing Pegasos algorithm ([59], `https://github.com/bruincui/Pegasos`), where the parameter C is selected from Ω. The mini-batch size is 1 and the maximum number of iterations is $2m$.
- `SVRG` SVM with the squared hinge soft-margin loss is addressed by employing `SVRG` algorithm ([60], `https://github.com/codes-kzhan/SVRG-1/blob/master/SVM/svm_SVRG.m`) with C selected from Ω. The mini-batch size is 1 and the number of "passes" is $S = 1$. The default epoch length is $2m$.
- `KATY` SVM with the squared hinge soft-margin loss is addressed by Katyusha algorithm ([61], `https://github.com/codes-kzhan/SVRG-1/blob/master/SVM/svm_Katyusha.m`) with all parameters selected the same as these for `SVRG`.

In addition, all other parameters of the above nine algorithms are set to their default values.

(c) Evaluation criteria. To evaluate classification performance, we report five evaluation criteria: the testing accuracy (`ACC`), the number of support vectors (`NSV`), the size of working set per iteration (`SWS/ITER`), the total number of iterations (`TNI`) and the `CPU` time (`CPU`). Let $\{(\mathbf{x}_j^{\rm test}, y_j^{\rm test}) : j = 1, \cdots, m_t\}$ be the testing samples data. The testing accuracy is defined as follows

$$\texttt{ACC} := 1 - \frac{1}{2m_t}\sum_{j=1}^{m_t}\left|\text{sign}(\langle \mathbf{w}^*, \mathbf{x}_j^{\rm test}\rangle + b^*) - y_j^{\rm test}\right|,$$

where $\text{sign}(\overline{a}) = 1$ if $\overline{a} > 0$ and $\text{sign}(\overline{a}) = -1$ otherwise, and $(\mathbf{w}^*, b^*)$ is a solution obtained by one solver. The accuracy measures the ability of a solver to correctly predict the class labels of new input samples. The higher `ACC` (or the smaller `NSV`, `SWS/ITER`, `TNI` or `CPU`) is, the better performance of a solver delivers.

5.1 Comparisons with Synthetic Data

For visualization, we first consider a two-dimensional example, where the features come from Gaussian distributions [14], [57]. One can observe that $\texttt{L}_{0/1}\texttt{ADMM}$ performs extraordinarily in terms of delivering a considerably small number of support vectors.

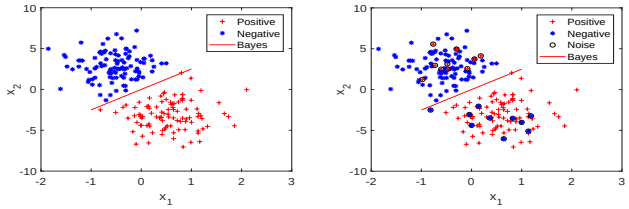
Example 5.1 (Synthetic data in $\mathbb{R}^2$ without outliers). In this example, m samples $\mathbf{x}_i, i \in \mathbb{N}_m$ with positive labels $y_i = +1$ are drawn from $N(\boldsymbol{\mu}_1, \Sigma_1)$ and samples $\mathbf{x}_i$ with negative labels $y_i = -1$ are drawn from $N(\boldsymbol{\mu}_2, \Sigma_2)$, where $\boldsymbol{\mu}_1 = [0.5, -3]^\top$, $\boldsymbol{\mu}_2 = [-0.5, 3]^\top$ and $\Sigma_1 = \Sigma_2 = \begin{bmatrix} 0.2 & 0 \\ 0 & 3 \end{bmatrix}$. We generate $2m$ samples with two classes having equal numbers, and then evenly split all samples into a training set and a testing set.

Data generated in this way has centralized features of each class. For this example, the corresponding Bayes classifier is $2.5x_1 - x_2 + 0 = 0$. We display Bayes classifier and 200 training samples in Figure 1 (a), where samples are no extra noises contaminated. We then add outliers on the data generated in Example 5.1 as follows.

Example 5.2 (Synthetic data in $\mathbb{R}^2$ with outliers). Firstly, $2m$ samples with two classes having equal numbers are generated as in Example 5.1. Then in each class, we randomly flip r percentage of labels. For instance, in m samples with positive labels $+1$, we change mr labels to -1. This means r percentage of $2m$ samples are flipped their labels, namely $2rm$ outliers are generated. Here r is the flapping ratio. Finally, the $2m$ samples are evenly split into a training set and a testing set. In Figure 1 (b), the training set with r=10% outliers are presented.

To solve these two examples, ten solvers are applied to calculate the classifier $w_1x_1 + w_2x_2 + b = 0$. Since data are generated randomly, to avoid randomness, we report average results of `ACC`, `NSV`, `SWS/ITER`, `TNI` and `CPU` over 10 times.

(d) Synthetic data without outliers. Ten solvers are applied to solve Example 5.1 with both the training and testing sample sizes being $m \in \{2000, 4000, \cdots, 10000\}$. Average



(a) (b)

Fig. 1: (a) A two dimensional training set with 200 samples. (b) Data in (a) but with r=10% outliers. Blue stars: sampling samples in class -1. Red crosses: sampling samples in class $+1$. Red dashed lines: the Bayes classifier.

results are reported in Table 1, where "$--$" represents that the results are not obtained if one solver takes time longer than two hour (denote "$> 2h$") or the required memory is out of the capacity of our laptop (denote "$**$"), and "3(8)" means the number of outer iterations (the average number of inner subproblem iterations). It can be clearly seen that all algorithms achieve desirable ACC and $L_{0/1}$ADMM gets slightly better ones. When it comes to NSV, the result is significant different. Obviously, LSVM, PSVM and LOGI take all samples as the support vectors, while HSVM, RSVM, SSVM, PEGA, SVRG and KATY have a small number of the support vectors. It is evidently that $L_{0/1}$ADMM uses a considerably small number of the support vectors. As for SWS/ITER and TNI, LSVM, PSVM, RSVM, SSVM and LOGI take all samples as the working set, while most of them use a small TNI except for PSVM. By contrast, $L_{0/1}$ADMM and others select a very small portion of samples as the working set, and $L_{0/1}$ADMM uses a small TNI (no more than 50 for all cases). Because of this, $L_{0/1}$ADMM consumes the shortest CPU time.

(e) Synthetic data with outliers. For Example 5.2, we fix $m = 5000, n = 2$ while alter the flapping ratio r from $\{0, 0.05, 0.1, 0.15, 0.2\}$ to see the robustness of each method to outliers. Average results are presented in Table 2. Apparently, the more outliers, the smaller ACC for each solver. There is no big difference of ACC generated by ten solvers. Again, $L_{0/1}$ADMM gets slightly better ACC, being more robust to outliers than the others. Similar observations to that in Table 1 can be seen for NSV, SWS/ITER and TNI. Moreover, the more outliers are added, the more examples become support vectors for HSVM, SSVM, PEGA, SVRG and KATY, and bigger values of TNI are generated by HSVM and PSVM. By contrast, $L_{0/1}$ADMM makes use of fewer support vectors, SWS/ITER and TNIwhen more outliers are added. Not surprisingly, $L_{0/1}$ADMM again runs the fastest.

5.2 Comparisons with Real Data

We now apply these solvers to deal with 14 real datasets. Their information are presented in Table 3, where the last six datasets have the testing data.

Example 5.3 (Real data without outliers). We perform 10-fold cross validation for the first eight datasets. Each one is randomly split into ten parts, with one part being used for testing and the rest being used for training. We then record average results to evaluate performance. In our experiments, all features are scaled to $[-1, 1]$.

Example 5.4 (Real data with outliers). To see the influence of the real data with outliers, we select six datasets from small sizes to moderate sizes in Table 3. They are col, aus, two, mus, adu and a6a. Same processes as in Example 5.3 are then applied into the first five datasets. Finally, r percentage of training and testing samples are randomly treated as outliers (i.e., their labels are flipped).

TABLE 1: Comparisons of 10 solvers for solving Example 5.1, where $L_{0/1}$ stands for $L_{0/1}$ADMM.

	ACC (%)									
m	$L_{0/1}$	HSVM	LSVM	PSVM	RSVM	SSVM	LOGI	PEGA	SVRG	KATY
2000	**97.05**	**97.05**	97.00	**97.05**	**97.05**	**97.05**	97.03	97.01	**97.05**	**97.05**
4000	**97.35**	97.25	97.30	97.30	97.33	97.32	97.25	97.26	97.33	**97.35**
6000	**97.33**	97.28	**97.33**	97.24	**97.33**	$--$	97.22	97.16	**97.33**	97.30
8000	**96.96**	96.91	96.89	96.91	**96.96**	$--$	**96.96**	96.93	96.94	**96.96**
10000	**97.20**	97.18	97.16	97.19	**97.20**	$--$	97.18	97.16	97.18	97.18
	NSV									
2000	**7**	187	2000	2000	96	146	2000	198	184	192
4000	**10**	301	4000	4000	141	289	4000	325	332	295
6000	**18**	439	6000	6000	201	$--$	6000	453	444	452
8000	**26**	571	8000	8000	223	$--$	8000	566	579	563
10000	**22**	658	10000	10000	240	$--$	10000	669	675	648
	SWS/ITER									
2000	22	2	2000	2000	2000	2000	2000	**1**	**1**	**1**
4000	31	2	4000	4000	4000	4000	4000	**1**	**1**	**1**
6000	35	2	6000	6000	6000	6000	6000	**1**	**1**	**1**
8000	38	2	8000	8000	8000	8000	8000	**1**	**1**	**1**
10000	46	2	10000	10000	10000	10000	10000	**1**	**1**	**1**
	TNI									
2000	20	259	14	1216	3(8)	2(13)	**8**	4000	4000	4000
4000	28	463	14	2325	3(16)	3(18)	**9**	8000	8000	8000
6000	34	639	15	3750	4(15)	$--$	**9**	12000	12000	12000
8000	40	772	16	5247	4(21)	$--$	**9**	16000	16000	16000
10000	47	961	16	6326	5(23)	$--$	**10**	20000	20000	20000
	CPU (seconds)									
2000	**0.002**	0.014	0.221	9.642	3.969	132.5	0.034	0.028	0.024	0.025
4000	**0.006**	0.022	0.626	67.58	16.29	2043	0.112	0.089	0.087	0.088
6000	**0.008**	0.036	1.200	209.9	31.44	$> 2h$	0.204	0.133	0.126	0.131
8000	**0.013**	0.069	2.342	493.2	65.25	$> 2h$	0.536	0.194	0.185	0.188
10000	**0.018**	0.094	3.951	775.3	124.7	$> 2h$	0.938	0.281	0.266	0.268

(f) Real data without outliers. The average results are recorded in Table 4, where "$> 3e5$" represents the number greater than 300000. It can be clearly seen that $L_{0/1}$ADMM outperforms the others in terms of the highest ACC, smallest NSV and shortest CPU for most datasets, and uses a small SWS/ITER and TNI. For instance, $L_{0/1}$ADMM predicts more than 90% samples correctly for col whilst HSVM and PSVM only get less than 80% correct predictions. Compared with those generated by the other nine solvers, NSV from $L_{0/1}$ADMM is relatively small. As for SWS/ITER, $L_{0/1}$ADMM takes a small samples as the working set, which testifies that our constructed working set strategy is very effective to reduce the cost of per iteration. As for TNI, $L_{0/1}$ADMM uses a few TNI compared with PSVM, PEGA, SVRG and KATY. For the computational speed, PEGA, SVRG and KATY present the advantage of CPU for dealing with small scale datasets. The $L_{0/1}$ADMM runs super fast for datasets in big sizes, 0.573 seconds v.s. 36.95 seconds by HSVM for data ijc. In addition, it only needs 14.26 seconds for the dataset hig with more than ten million samples. Overall, it seems that the bigger m is, the more evident the advantage of $L_{0/1}$ADMM becomes.

(g) Real data with outliers. Finally, we would like to see the robustness of each solver to the outliers for real

TABLE 2: Comparisons of 10 solvers for solving Example 5.2

r	$L_{0/1}$	HSVM	LSVM	PSVM	RSVM	SSVM	LOGI	PEGA	SVRG	KATY
					ACC (%)					
0.00	**97.16**	97.08	97.10	**97.16**	**97.16**	97.12	97.08	97.03	**97.16**	**97.16**
0.05	**92.65**	92.46	92.50	92.60	**92.65**	92.57	92.58	92.54	92.30	92.35
0.10	**87.98**	87.78	87.78	87.90	87.90	87.90	87.70	87.68	87.46	87.45
0.15	**83.06**	82.86	82.80	82.98	**83.06**	83.04	82.93	82.98	82.88	82.88
0.20	**78.30**	78.16	78.12	78.28	78.28	78.20	78.16	78.21	78.17	78.18
					NSV					
0.00	**21**	364	5000	5000	184	329	5000	372	359	357
0.05	**20**	947	5000	5000	175	874	5000	942	953	945
0.10	**17**	1385	5000	5000	170	1015	5000	1365	1373	1389
0.15	**16**	1795	5000	5000	161	1657	5000	1790	1781	1792
0.20	**13**	2160	5000	5000	137	1989	5000	2177	2175	2187
					SWS/ITER					
0.00	34	2	5000	5000	5000	5000	5000	**1**	**1**	**1**
0.05	31	2	5000	5000	5000	5000	5000	**1**	**1**	**1**
0.10	30	2	5000	5000	5000	5000	5000	**1**	**1**	**1**
0.15	28	2	5000	5000	5000	5000	5000	**1**	**1**	**1**
0.20	27	2	5000	5000	5000	5000	5000	**1**	**1**	**1**
					TNI					
0.00	32	584	15	3042	3(25)	3(21)	**9**	10000	10000	10000
0.05	30	3726	15	3126	4(18)	3(21)	**9**	10000	10000	10000
0.10	29	5128	15	3268	4(17)	3(21)	**9**	10000	10000	10000
0.15	26	8423	15	3373	5(13)	3(21)	**9**	10000	10000	10000
0.20	25	10776	15	3443	5(13)	3(21)	**9**	10000	10000	10000
					CPU (seconds)					
0.00	**0.008**	0.027	0.801	93.11	22.53	4047	0.149	0.117	0.108	0.112
0.05	**0.008**	0.075	0.823	101.3	20.99	4069	0.131	0.119	0.114	0.115
0.10	**0.006**	0.123	0.853	105.4	19.43	4084	0.147	0.118	0.111	0.112
0.15	**0.005**	0.172	0.885	108.3	18.96	4092	0.152	0.118	0.110	0.111
0.20	**0.005**	0.236	0.898	110.6	18.41	4094	0.165	0.119	0.115	0.116

TABLE 3: Descriptions of 14 real datasets.

Datasets	Training data m	Testing data m_t	Features n
Colon-cancer (col)	62	0	2000
Australian (aus)	690	0	14
Two-norm (two)	7400	0	20
Mushrooms (mus)	8124	0	112
Adult (adu)	17887	0	13
Covtype.binaty (cov)	581012	0	54
SUSY (sus)	5000000	0	18
HIGGS (hig)	11000000	0	28
Lekemia (lek)	38	34	7129
Splice (spl)	1000	2175	60
A6a (a6a)	11220	21341	123
W6a (w6a)	17188	32561	300
W8a (w8a)	49749	14951	300
ijcnn1 (ijc)	49990	91701	22

datasets in Example 5.4. Again we alter the flapping ratio r from $\{0.01, 0.02, \cdots, 0.1\}$. It is shown in Table 4 that SSVM takes too long time for datasets: two, mus, adu and a6a. Therefore, its results related to these datasets are omitted. All lines of ACC shown in Figure 2 decline with r ascending, and $L_{0/1}$ADMM achieves the highest ACC. As for NSV in Figure 3, LSVM, PSVM and LOGI always treat all samples as support vectors. HSVM, SSVM, PEGA, SVRG and KATY increase NSV with the rising of r. Lines from $L_{0/1}$ADMM and RSVM either decline or stabilize at a level with the rising of r, which means they are quite robust to r, namely robust to the outliers. What is more, $L_{0/1}$ADMM always renders the fewest NSV. As for SWS/ITER in Figure 4, with the

TABLE 4: Comparisons of 10 solvers for solving Example 5.3

Data	$L_{0/1}$	HSVM	LSVM	PSVM	RSVM	SSVM	LOGI	PEGA	SVRG	KATY
					ACC (%)					
col	**90.23**	64.52	85.48	77.69	89.68	85.87	86.74	89.68	89.68	89.68
aus	**86.23**	85.51	85.80	85.80	86.02	85.98	86.18	86.04	86.18	**86.23**
lek	**82.35**	58.82	79.41	58.82	76.47	**82.35**	**82.35**	**82.35**	**82.35**	**82.35**
spl	85.52	**88.97**	85.75	85.52	85.47	85.47	85.15	84.18	85.44	85.33
two	**98.37**	98.02	97.97	97.97	98.24	−−	97.78	98.10	**98.37**	98.24
mus	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	−−	**100.0**	**100.0**	**100.0**	**100.0**
adu	**83.90**	83.29	83.01	83.07	83.79	−−	82.95	83.29	83.34	**83.90**
a6a	**84.90**	84.18	84.55	84.69	84.72	−−	84.76	84.36	84.72	84.78
w6a	**97.93**	97.21	97.58	97.21	97.86	−−	95.13	97.24	97.61	97.57
w8a	**98.54**	98.27	−−	−−	−−	−−	−−	97.43	97.57	97.59
ijc	**94.33**	92.73	−−	−−	−−	−−	−−	93.49	93.35	93.56
cov	**71.79**	−−	−−	−−	−−	−−	−−	68.93	69.83	69.77
sus	**67.58**	−−	−−	−−	−−	−−	−−	64.28	65.62	65.86
hig	**65.21**	−−	−−	−−	−−	−−	−−	58.12	59.13	59.46
					NSV					
col	**34**	46	54	54	38	40	54	46	45	46
aus	**24**	203	621	621	89	177	621	198	195	202
lek	**26**	31	38	38	29	31	38	33	34	31
spl	**70**	607	1000	1000	87	332	1000	632	615	612
two	**30**	758	6600	6600	108	−−	6600	783	775	788
mus	**135**	550	7311	7311	506	−−	7311	578	575	568
adu	**113**	6379	16098	16098	1247	−−	16098	6407	6386	6394
a6a	**370**	4346	11220	11220	1247	−−	11220	4562	4575	4582
w6a	**429**	1128	17188	17188	946	−−	17188	1146	1152	1138
w8a	**867**	2857	−−	−−	−−	−−	−−	2582	2579	2561
ijc	**215**	8508	−−	−−	−−	−−	−−	8535	8612	8608
cov	**137**	−−	−−	−−	−−	−−	−−	>3e5	>3e5	>3e5
sus	**730**	−−	−−	−−	−−	−−	−−	>2e6	>2e6	>2e6
hig	**1338**	−−	−−	−−	−−	−−	−−	>5e6	>5e6	>5e6
					SWS/ITER					
col	37	2	54	54	54	54	54	**1**	**1**	**1**
aus	66	2	621	621	621	621	621	**1**	**1**	**1**
lek	29	2	38	38	38	38	38	**1**	**1**	**1**
spl	94	2	1000	1000	1000	1000	1000	**1**	**1**	**1**
two	136	2	6600	6600	6600	−−	6600	**1**	**1**	**1**
mus	772	2	7311	7311	7311	−−	7311	**1**	**1**	**1**
adu	1105	2	16098	16098	16098	−−	16098	**1**	**1**	**1**
a6a	569	2	11220	11220	11220	−−	11220	**1**	**1**	**1**
w6a	656	2	17188	17188	17188	−−	17188	**1**	**1**	**1**
w8a	1284	2	−−	−−	−−	−−	−−	**1**	**1**	**1**
ijc	829	2	−−	−−	−−	−−	−−	**1**	**1**	**1**
cov	1520	−−	−−	−−	−−	−−	−−	**1**	**1**	**1**
sus	2814	−−	−−	−−	−−	−−	−−	**1**	**1**	**1**
hig	3225	−−	−−	−−	−−	−−	−−	**1**	**1**	**1**
					TNI					
col	30	41	**2**	31	2(2)	2(4)	4	108	108	108
aus	25	423	17	869	2(7)	3(26)	**6**	1242	1242	1242
lek	18	89	**2**	42	2(2)	3(17)	25	76	76	76
spl	63	595	28	1276	2(9)	4(28)	**9**	2000	2000	2000
two	50	660	75	3417	4(11)	−−	**12**	13200	13200	13200
mus	21	1623	106	3685	4(12)	−−	**18**	14622	14622	14622
adu	26	4766	157	7720	5(21)	−−	**15**	32196	32196	32196
a6a	183	3032	289	6873	5(27)	−−	**16**	22440	22440	22440
w6a	121	1450	404	14417	7(32)	−−	**28**	34376	34376	34376
w8a	**195**	8124	−−	−−	−−	−−	−−	99498	99498	99498
ijc	**146**	6681	−−	−−	−−	−−	−−	99980	99980	99980
cov	**103**	−−	−−	−−	−−	−−	−−	$1.05e^6$	$1.05e^6$	$1.05e^6$
sus	**117**	−−	−−	−−	−−	−−	−−	$9.0e^6$	$9.0e^6$	$9.0e^6$
hig	**124**	−−	−−	−−	−−	−−	−−	$1.98e^7$	$1.98e^7$	$1.98e^7$
					CPU (seconds)					
col	0.021	0.009	**0.001**	0.010	0.003	1.488	0.182	0.015	0.012	0.014
aus	0.005	0.014	0.033	0.874	0.650	87.23	0.021	**0.004**	**0.004**	**0.004**
lek	0.072	0.057	**0.004**	0.010	0.008	54.36	36.10	0.029	0.024	0.026
spl	0.043	0.117	0.083	7.976	0.631	384.2	0.151	0.036	**0.032**	0.033
two	**0.054**	0.265	2.506	516.7	139.2	$> 2h$	1.591	0.171	0.164	0.166
mus	**0.074**	0.997	3.419	769.5	153.4	$> 2h$	6.942	0.422	0.412	0.416
adu	**0.576**	3.775	24.58	1633.4	1013.2	$> 2h$	5.032	0.775	0.732	0.744
a6a	**0.172**	4.405	40.64	1472.5	1037.3	$> 2h$	6.046	1.083	1.025	1.031
w6a	**0.226**	1.532	170.9	5947.2	2747.4	$> 2h$	41.21	1.314	1.186	1.232
w8a	**2.576**	64.33	**	**	$> 2h$	$> 2h$	**	4.863	4.227	4.316
ijc	**0.573**	36.95	**	**	$> 2h$	$> 2h$	**	1.526	1.247	1.316
cov	**3.870**	**	**	**	**	**	**	14.37	13.88	13.91
sus	**10.38**	**	**	**	**	**	**	137.6	132.4	133.7
hig	**14.26**	**	**	**	**	**	**	281.3	269.5	270.1

rising of r, $\mathtt{L_{0/1}ADMM}$ stabilizes at a level for all datasets. As for TNI in Figure 5, all algorithms no big difference with the ascending of r except for HSVM. For the computational speed, as demonstrated in Figure 6, $\mathtt{L_{0/1}ADMM}$ outperforms the others for all datasets except for col and aus which have a very small size.

6 CONCLUSION

In this paper, we have explored an ideal soft-margin SVM model: $L_{0/1}$-SVM, which well captures the nature of the binary classification and guarantees a fewer number of support vectors than the other soft-margin SVM models. Despite the discreteness of the $L_{0/1}$-SVM, the establishment of the optimality theory, associated with the P-stationary point, makes it tractable numerically. Based on the idea of $L_{0/1}$ support vectors inspired by the P-stationary point, a working set was cast and integrated into the proximal ADMM, which tremendously speeds up the whole computation and reduces the number of support vectors. Consequently, the proposed method performed exceptionally well with fewer support vectors and faster computational speed, especially for datasets on large scales.

We feel that the established methodology and techniques might be able to extend to process the nonlinear kernel SVMs [62]–[64] and problems from perception learning [4] and deep learning [5]. We leave these as future research.

ACKNOWLEDGEMENTS

The authors would like to thank the associate editor and three anonymous referees for their constructive comments, which have significantly improved the quality of the paper. This work is supported by the National Natural Science Foundation of China (11971052, 11926348-9, 61866010, 11871183), and the Natural Science Foundation of Hainan Province (120RC449).

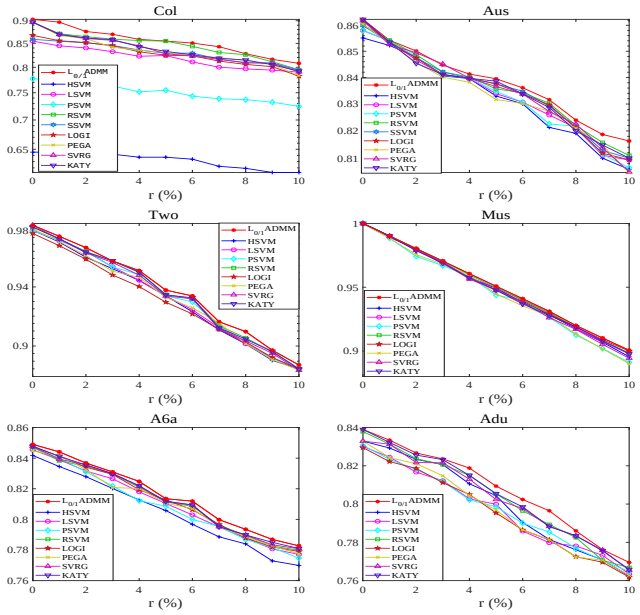


Fig. 2: ACC vs. r of all solvers for solving six datasets.

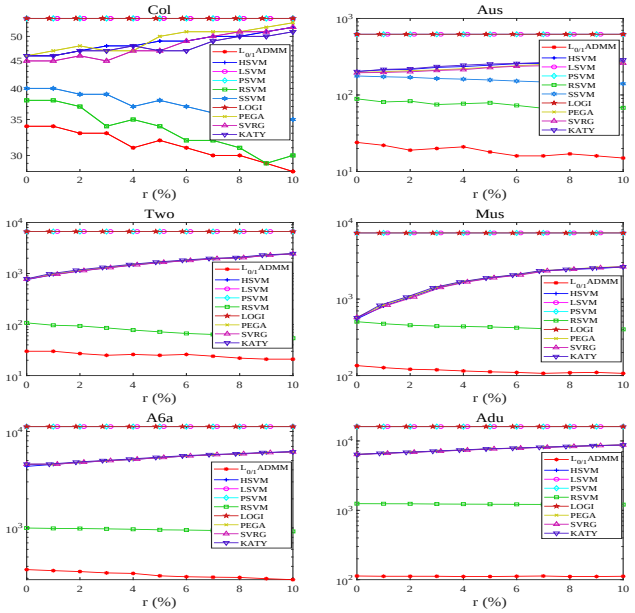


Fig. 3: NSV vs. r of all solvers for solving six datasets.

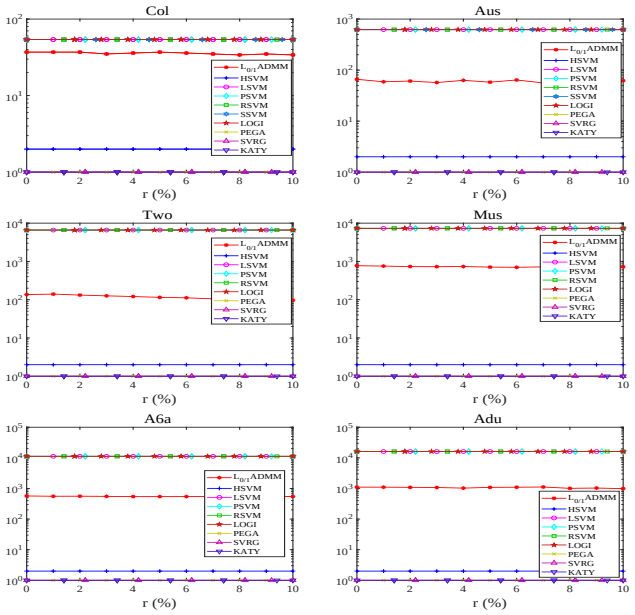


Fig. 4: SWS/ITER vs. r of all solvers for solving six datasets.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support vector networks", *Mach. Learn.*, vol. 20, no. 3, pp. 273-297, 1995.

[2] J. P. Brooks, "Support vector machines with the ramp loss and the hard margin loss", *Oper. Res.*, vol. 59, no. 2, pp. 467-479, 2011.

[3] Y. L. Feng, Y. N. Yang, X. L. Huang, S. Mehrkanoon, and J. A. K. Suykens, "Robust support vector machines for classification with nonconvex and smooth losses", *Neural Comput.*, vol. 28, no. 6, pp. 1217-1247, 2016.

[4] L. Li and H. T. Lin, "Optimizing 0/1 loss for perceptrons by random coordinate descent", *in Proc. IEEE Int. Joint Conf. Neural Netw.*, 2007, pp. 649-654.

[5] I. Goodfellow, B. Yoshua, and C. Aaron, "Deep learning",*MIT press*, 2016.

[6] W. H. Hu, G. Niu, I. Sato, and M. Sugiyama, "Does distributionally robust supervised learning give robust classifiers?", *in Proc. 35th Int. Conf. Mach. Learn.*, pp. 2029-2037, 2018.

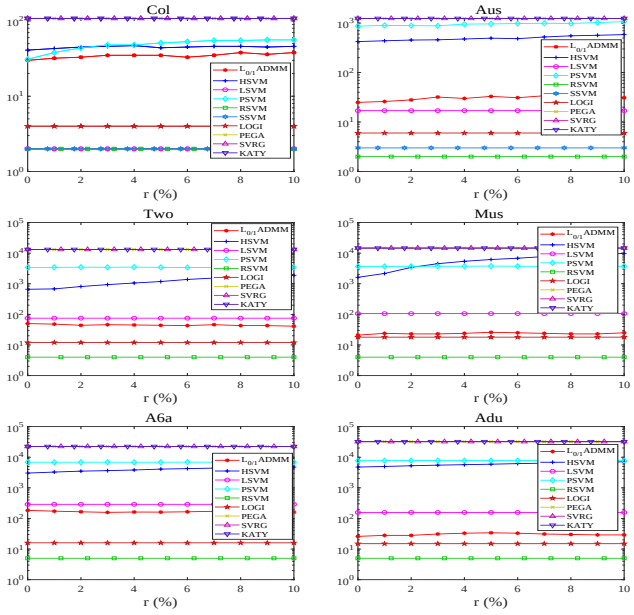


Fig. 5: TNI vs. r of all solvers for solving six datasets.

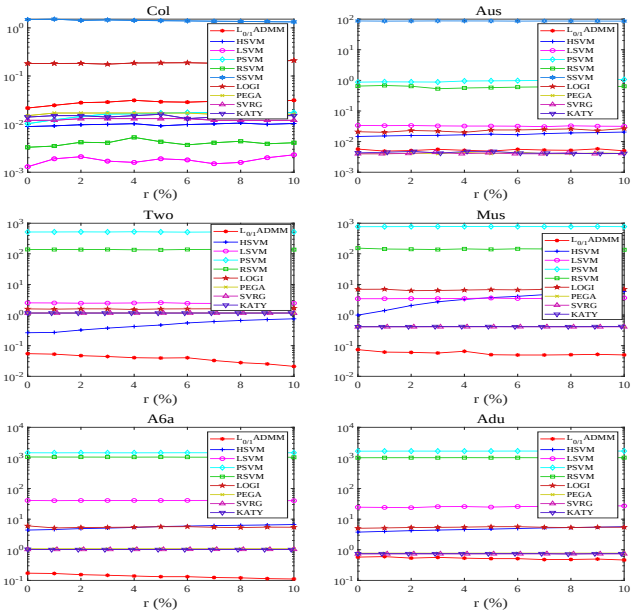


Fig. 6: CPU vs. r of all solvers for solving six datasets.

[7] B. K. Natarajan," Sparse approximate solutions to linear systems", *SIAM J. Comput.*, vol. 24, no. 2, pp. 227-234, 1995.

[8] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems", *Theor. Comput. Sci.*, vol. 209, no. 1, pp. 237-260, 1998.

[9] A. Cotter, S. Shalev-Shwartz, and N. Srebro, "Learning optimally sparse support vector machines", *in Proc. Int. Conf. Mach. Learn., 2013,* pp. 266-274.

[10] H. Masnadi-Shirazi and N. Vasconcelos, "On the design of loss functions for classification: theory, robustness to outliers, and savageboost", *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 1049-1056, 2009.

[11] Y. M. Lyu and W. I. Tsang, "Curriculum loss: robust learning and generalization against label corruption", *arXiv preprint arXiv:1905.10045*, 2019.

[12] B. Schoelkopf and A. J. Smola," Learning with kernels", *MIT Press,* 2002.

[13] V. Jumutc, X. Huang, and J. A. K. Suykens, "Fixed-size pegasos for hinge and pinball loss SVM", *in Proc. IEEE Int. Joint Conf. Neural Netw.,*pp. 1-7, 2013.

[14] X. Huang, L. Shi, and J. A. K. Suykens, "Support vector machine classifier with pinball loss", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 5, pp. 984-997, 2014.

[15] L. Wang, J. Zhu, and H. Zou, "Hybrid huberized support vector machines for microarray classification", *Bioinformatics,* vol. 24, no. 3, pp. 412-419, 2008.

[16] Y. Xu, I. Akrotirianakis, and A. Chakraborty, "Proximal gradient method for huberized support vector machine", *Pattern Anal. Appl.*, vol. 19, no. 4, pp. 989-1005, 2016.

[17] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", *Neural Process. Lett.*, vol. 9, no. 3, pp. 293-300, 1999.

[18] X. Yang, L. Tan, and L. F. He, "A robust least squares support vector machine for regression and classification with noise", *Neurocomputing,* vol. 140, pp. 41-52, 2014.

[19] T. Zhang and F. J. Oles, "Text categorization based on regularized linear classification methods", *Information Retrieval,* vol. 4, no. 1, pp. 5-31, 2008.

[20] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting", *Ann. Stat.*, vol. 28, no. 2, pp. 337-374, 2000.

[21] P. L. Bartlett and H. W. Marten, "Classification with a reject option using a hinge loss", *J. Mach. Learn. Res.* vol. 9, no. 8, pp. 1823-1840, 2008.

[22] P. L. Bartlett, M. I. Jordan, and J. D. Mcauliffe, "Large margin classifiers: convex loss, low noise, and convergence rates", *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 1173-1180, 2004.

[23] P. L. Bartlett, M. I. Jordan, and J. D. Mcauliffe, "Convexity, classification, and risk bounds", *J. Am. Stat. Assoc.*, vol. 101, no. 473, pp. 138-156, 2006.

[24] J. H. Friedman, "On bias, variance, 0/1-loss, and the curse-of-dimensionality", *Data Min. Knowl. Discov.*, vol. 1, no. 1, pp. 55-77, 1997.

[25] L. Mason, P. L. Bartlett, and J. Baxter, "Improved generalization through explicit optimization of margins," *Mach. Learn.*, vol. 38, no. 3, pp. 243-255, 2000.

[26] F. Perez-Cruz, A. Navia-Vazquez, A. R. Figueiras-Vidal, and A. Artes-Rodriguez, "Empirical risk minimization for support vector classifiers", *IEEE Trans. Neural Netw.*, vol. 14, no. 2, pp. 296-303, 2003.

[27] R. Collobert, F. Sinz, J. Weston, L. Bottou, "Trading convexity for scalability", *in Proc. 23th Int. Conf. Mach. Learn.*, 2006, pp. 201-208.

[28] X. Huang, L. Shi, and J. A. K. Suykens, "Ramp loss linear programming support vector machine", *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2185-2211, 2014.

[29] X. Shen, L. F. Niu, Z. Qi, and Y. J. Tian, "Support vector machine classifier with truncated pinball loss", *Pattern Recognit.*, vol. 68, pp. 199-210, 2017.

[30] L. M. Yang and H. G. Dong, "Support vector machine with truncated pinball loss and its application in pattern recognition", *Chemometrics Intell. Lab. Syst.*, vol. 177, pp. 89-99, 2018.

[31] F. Perez-Cruz, A. Navia-Vazquez, P. L. Alarcon-Diana, and A. Artes-Rodriguez, "Support vector classifier with hyperbolic tangent penalty function", *in Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 3458-3461, 2000.

[32] L. Wang, H. D. Jia, and J. Li, "Training robust support vector machine with smooth ramp loss in the primal space", *Neurocomputing*, vol. 71, no. 13, pp. 3020-2025, 2008.

[33] I. Steinwart and A. Christmann, "Support vector machines", *New York: Springer,* 2008.

[34] S. Y. Park and Y. F. Liu, "Robust penalized logistic regression with truncated loss functions", *Canadian Journal of Statistics*, vol. 39, no. 2, pp. 300-323, 2011.

[35] D. L. Liu, Y. Shi, Y. J. Tian, and X. K. Huang, "Ramp loss least squares support vector machine", *J. Comput. Sci.*, vol. 14, pp. 61-68, 2016.

[36] I. Steinwart and N. Christianini, "Sparseness of support vector machines", *J. Mach. Learn. Res.*, vol. 4, no. 6, pp. 1071-1105, 2004.

[37] S. Ertekin, L. Bottou, and C. L. Giles, "Nonconvex online support vector machines", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 4, pp. 368-381, 2010.

[38] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations", *J. Fourier Anal.Appli.*, vol. 14, no. 5-6, pp. 629-654, 2008.

[39] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing", *Appl. Comput. Harmonic Anal.*, vol. 27, no. 3, pp. 265-274, 2009.

[40] Z. Lu and Y. Zhang, "Sparse approximation via penalty decomposition methods", *SIAM J. Optim.*, vol. 23, no. 4, pp. 2448-2478, 2013.

[41] Z. S. Lu, "Iterative reweighted minimization methods for l_p-regularized unconstrained nonlinear programming", *Math. Program.*, vol. 147, no.1-2, pp. 277-307, 2014.

[42] A. Beck and N. Hallak, "Proximal mapping for symmetric penalty and sparsity", *SIAM J. Optim.*, vol. 28, no. 1, pp. 496-527, 2018.

[43] H. Zhang, L. L. Pan, and N. H. Xiu, "Optimality conditions for locally Lipschitz optimization with l_0-regularization", *Optim. Lett.*, DOI: 10.1007/s11590-020-01579-y, 2020.

[44] M. Fazel, T. K. Pong, D. F. Sun, and P. Tseng, "Hankel matrix rank minimization with applications to system identification and realization", *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 3, pp. 946-977, 2013.

[45] M. Li, D. F. Sun, and K. C. Toh, "A majorized ADMM with indefinite proximal terms for linearly constrained convex composite optimization", *SIAM J. Optim.*, vol. 26, no. 2, pp. 922-950, 2016.

[46] X. Chang, S. Liu, P. Zhao, and D. Song, "A generalization of linearized alternating direction method of multipliers for solving two-block separable convex programming", *J. Comput. Appl. Math.*, vol. 357, no. 2, pp. 251-272, 2019.

[47] G. Golub and C. F. Van-Loan, "Matrix computations", *Johns Hopkins University Press*, 1996.

[48] Y. Wang, W. T. Yin, and J. S. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization", *J. Sci. Comput.*, vol. 78, no. 1, pp. 29-63, 2019.

[49] G. Y. Li and T. K. Pong, "Global convergence of splitting methods for nonconvex composite optimization", *SIAM J. Optim.*, vol. 25, no. 4, pp. 2434-2460, 2015.

[50] M. Hong, Z. Luo, and M. Razaviyayn, "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems", *SIAM J. Optim.*, vol. 26, no. 1, pp. 337-364, 2016.

[51] R. I. Bot and D. K. Nguyen, "The proximal alternating direction method of multipliers in the nonconvex setting: convergence analysis and rates", *Math. Oper. Res.*, vol. 45, no. 2, pp. 682-712, 2020.

[52] F. P. Nie, Y. Z. Huang, X. Q. Wang, and H. Huang, "New primal SVM solver with linear computational cost for big data classifications", *in Proc. 31th Int. Conf. Mach. Learn.*, pp. 505-513, 2014.

[53] L. Guan, L. B. Qiao, D. S. Li, T. Sun, K. S. Ge, and X. C. Lu, "An efficient ADMM-based algorithm to nonconvex penalized support vector machines", *in Proc. Int. Conf. Data Mining Workshops*, 1209-1216, 2018.

[54] J. Z. Zhang, H. Z. Lin, S. Jegelka, A. Jadbabaie, and S. Sra, "On complexity of finding stationary points of nonsmooth nonconvex functions", *arXiv preprint arXiv:2002.04130*, 2020.

[55] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines", *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27, 2011.

[56] K. Pelckmans, J. A. K. Suykens, T. V. Gestel, J. D. Brabanter, L. Lukas, B. Hamers, B. D. Moor, and J. Vandewalle, "LSSVM lab: a matlab/c toolbox for least squares support vector machines", *Tutorial. KULeuven-ESAT. Leuven, Belgium*, vol. 142, pp. 1-2, 2002.

[57] X. Huang, L. Shi, and J. A. K. Suykens, "Solution path for pin-SVM classifiers with positive and negative τ values", *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 7, pp. 1584-1593, 2016.

[58] T. P. Minka, "A comparison of numerical optimizers for logistic regression", *Available on http://yaroslavvb.com/papers/minka-comparison.pdf*, 2003.

[59] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter, "Pegasos: primal estimated sub-gradient solver for SVM", *Math. Program.*, vol. 127, no. 1, pp. 3-30, 2011.

[60] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction", *in Proc. Int. Conf. Neural Inf. Process. Syst.*, pp. 315-323, 2013.

[61] Z. Allen-Zhu, "Katyusha: the first direct acceleration of stochastic gradient methods", *J. Mach. Learn. Res.*, vol. 18, no. 221, pp. 1-51, 2018.

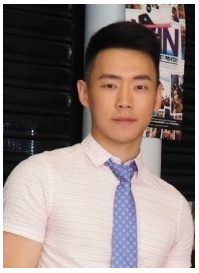
[62] H. V. Nguyen and F. Porikli, "Support vector shape: a classifier-based shape representation", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 4, pp. 970-982, 2012.

[63] Y. Tang, "Deep learning using linear support vector machines", *arXiv preprint arXiv:1306.0239*, 2013.

[64] B. Hong, W. Z. Zhang, W. Liu, J. P. Ye, D. Cai, X. f. He, and J. Wang, "Scaling up sparse support vector machines by simultaneous feature and sample reduction", *J. Mach. Learn. Res.*, vol. 20, no. 121, pp. 1-39, 2019.

Huajun Wang received his M.Sc. degree in Department of Mathematics from Guilin University of Electronic Technology, China, in 2017. He is currently a Ph.D. candidate of Department of Applied Mathematics at the Beijing Jiaotong University, China. His current research interests include large-scale classification optimization problems, machine learning, 0-1 loss optimization and numerical computing.

Yuanhai Shao received his B.Sc. degree in College of Mathematics from Jilin University, and received Ph.D. degree in College of Science from China Agricultural University, China, in 2006 and 2011, respectively. Currently, he is a professor at the Management School, Hainan University. His research interests include optimization methods, machine learning, and data mining. He has published over 100 refereed papers.



Shenglong Zhou received the B.Sc. degree in information and computing science in 2011 and the M.Sc. degree in operational research in 2014 from Beijing Jiaotong University, China, and the Ph.D. degree in operational research in 2018 from the University of Southampton, the United Kingdom, where he was a Research Fellow and Teaching Fellow from 2017 to 2021. He is currently a Research Associate at the Department of EEE, Imperial College London. His research interests include the theory and methods of optimization in the fields of sparse, low-rank matrix and bilevel optimization.

Ce Zhang received the B.Sc. degree in information and computing science and the M.Sc. degree in operational research from Beijing Jiaotong University, Beijing, China, in 2016 and 2019, respectively. He research interests include optimization methods, machine learning and applications in data and image processing.

Naihua Xiu received the B.Sc. degree in mathematics from Hebei Normal University, Shijiazhuang, China, in 1982, and the Ph.D. degree in operational research and optimal control from the Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing, China, in 1997. From 1997 to 1999, he was a Chinese Post-Doctoral Fellow with Beijing Jiaotong University, Beijing, where he was an Associate Professor in 1999 and has been a Professor in operational research since 2001. He was also a Research Fellow with the City University of Hong Kong, Hong Kong, from 2000 to 2002 and a Visiting Scholar with the University of Waterloo, Waterloo, ON, Canada, from 2006 to 2007.

His current research interests include machine learning, mathematical optimization, mathematics of operations research, and complementarity problems and variational inequalities. Dr. Xiu is the 9-10th Vice President of the Operations Research Society of China, and also serves as a member of Editorial Board for several journals such as Acta Mathematicae Applicatae Sinica, OR Transactions, Operations Research and Management, and Journal of the Operations Research Society of China.

Support Vector Machine Classifier via $L_{0/1}$ Soft-Margin Loss

Huajun Wang, Yuanhai Shao, Shenglong Zhou, Ce Zhang and Naihua Xiu*

Supplementary Material

The supplementary file covers technical lemmas and proofs.

S.1 Proof of Theorem 3.1

From (8), one can easily check that

$$\min_{\mathbf{w}\in\mathbb{R}^n, b\in\mathbb{R}} f(\mathbf{w};b) \le f(\mathbf{1};b) < n^2 + Cm < +\infty.$$

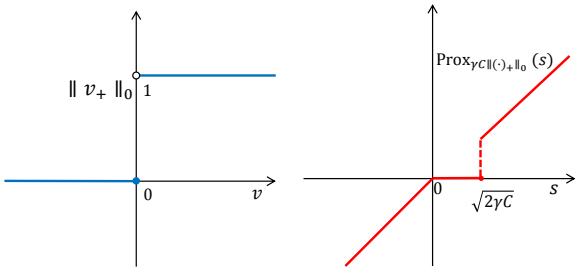
Denote

$$S := \{(\mathbf{w};b) \in \mathbb{R}^n \times \mathcal{I} : f(\mathbf{w};b) < n^2 + Cm\}.$$

Clearly, $S \neq \emptyset$ due to $(\mathbf{1};b) \in S$. For any $(\mathbf{w};b) \in S$, by

$$Cm + n^2 > f(\mathbf{w};b) \ge \|\mathbf{w}\|^2/2.$$

So $\mathbf{w}$ is bounded. Since $b \in \mathcal{I}$, it is also bounded. It follows from [1, Theorem 4.10] and the lower semicontinuity of f that the global optimal solution to (8) exists and the solution set is bounded, which completes the proof. □



(a) (b)

Fig. S.1: (a) The blue line (including the blue original) is the value of the function $\|v_+\|_0$. (b) the red line denotes the $\ell_{0/1}$ proximal operator.

S.2 $\ell_{0/1}$ and $L_{0/1}$ Proximal Operators

The proximal operators for nonconvex and continuous functions have been well studied, see, e.g. [2]–[4]. The proximal operators for zero norm $\|(\cdot)\|_0$ and the related functions have also been studied, see, e.g. [5]–[10]. Here, we investigate the proximal operator for the soft-margin loss function defined by (4')

$$\ell_{0/1}(v) = \begin{cases} 1, & v > 0, \\ 0, & v \le 0, \end{cases}$$

and its extension, which are footstone of optimality theory for the $L_{0/1}$-SVM. From the definition (7) of $\|(\cdot)_+\|_0$, we know that $\ell_{0/1}(v) = \|v_+\|_0$ (see, Figure S.1 (a)). First, we introduce the definition of $\ell_{0/1}$ proximal operator in one dimensional case.

Definition S.1 ***($\ell_{0/1}$ proximal operator).*** For any given $\gamma, C > 0$ and $s \in \mathbb{R}$, the proximal operator of $\|(v)_+\|_0$ (dubbed as $\ell_{0/1}$ proximal operator) is defined by

$$\mathrm{Prox}_{\gamma C\|(\cdot)_+\|_0}(s) = \arg\min_{v\in\mathbb{R}} C\|v_+\|_0 + \frac{1}{2\gamma}(v-s)^2. \tag{S1}$$

The following lemma states that the $\ell_{0/1}$ proximal operator admits a closed form solution.

Lemma S.1 (Solution to $\ell_{0/1}$ proximal operator). For any given $\gamma, C > 0$, the solution to $\ell_{0/1}$ proximal operator at $s \in \mathbb{R}$ is given by

$$\mathrm{Prox}_{\gamma C\|(\cdot)_+\|_0}(s) := \begin{cases} 0, & 0 < s < \sqrt{2\gamma C}, \\ 0 \text{ or } s, & s = \sqrt{2\gamma C}, \\ s, & s > \sqrt{2\gamma C} \text{ or } s \le 0. \end{cases} \tag{S2}$$

Proof. It follows from (S1) that

$$\mathrm{Prox}_{\gamma C\|(\cdot)_+\|_0}(s) = \arg\min_{v\in\mathbb{R}} \gamma C\|v_+\|_0 + \frac{1}{2}(v-s)^2.$$

Let $\phi(v) := \gamma C\|v_+\|_0 + (v-s)^2/2$. Since $\phi(v) = \gamma C + (v-s)^2/2 =: \phi_1(v)$ for $v > 0$ and $\phi(v) = (v-s)^2/2 =: \phi_2(v)$ for $v < 0$ are strongly convex and twice continuously differentiable, the unique minimal values of $\phi_1(v)$ and $\phi_2(v)$ are both attained at $v = s$. Moreover, it follows from $\phi(v) := (v-s)^2/2 =: \phi_3(0)$ for $v = 0$ that $\phi_3(0) = s^2/2$. The rest part is to compare the three values $\phi_1(s)$ with $s > 0$, $\phi_2(s)$ with $s < 0$ and $\phi_3(0)$:
(i) As $s > \sqrt{2\gamma C} \Leftrightarrow \phi_3(0) > \phi_1(s)$ and $\phi_2(s) > \phi_1(s)$, the minimal value of the $\phi(v)$ is achieved at $v = s$.
(ii) As $0 < s < \sqrt{2\gamma C} \Leftrightarrow \phi_1(s) > \phi_3(0)$ and $\phi_2(s) > \phi_3(0)$, it follows $v = 0$.
(iii) As $s = 0 \Leftrightarrow \phi_1(s) > \phi_3(0)$ and $\phi_2(s) > \phi_3(0)$, it is easy to check that $v = s$.
(iv) As $s < 0 \Leftrightarrow \phi_1(s) > \phi_2(s)$ and $\phi_3(0) > \phi_2(s)$, it is easy

to verify that $v = s$.
(v) As $s = \sqrt{2\gamma C} \Leftrightarrow \phi_2(s) > \phi_1(s) = \phi_3(0)$, then $v = 0$ or s. Thus, we have (S2), which completes the proof. □

It is worth mentioning that the solution to $\ell_{0/1}$ proximal operator may not be unique if $s = \sqrt{2\gamma C}$ in (S2). To guarantee the uniqueness, hereafter, we always choose the $\ell_{0/1}$ proximal operator to be zero if it is not unique. Because of this, the solution to $\ell_{0/1}$ proximal operator is rewritten as

$$\text{Prox}_{\gamma C\|(\cdot)_+\|_0}(s) := \begin{cases} 0, & 0 < s \le \sqrt{2\gamma C}, \\ s, & s > \sqrt{2\gamma C} \text{ or } s \le 0, \end{cases} \tag{S3}$$

see Figure S.1 (b).

Based on the one-dimensional case, we give the definition of the proximal operator of $\|\mathbf{v}_+\|_0$ in Multi-dimensional setting.

Definition S.2 ($L_{0/1}$ proximal operator). For any given $\gamma, C > 0$ and $\mathbf{s} \in \mathbb{R}^m$, the proximal operator of $\|\mathbf{v}_+\|_0$ (dubbed as $L_{0/1}$ proximal operator) is defined by

$$\text{Prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{s}) = \arg\min_{\mathbf{v}\in\mathbb{R}^m} C\|\mathbf{v}_+\|_0 + \frac{1}{2\gamma}\|\mathbf{v}-\mathbf{s}\|^2. \tag{S4}$$

The following lemma states that the $L_{0/1}$ proximal operator admits a closed form solution.

Lemma S.2 (Solution to $L_{0/1}$ proximal operator). For any given $\gamma, C > 0$, the solution to $L_{0/1}$ proximal operator at $\mathbf{s} = (s_1, s_2, ..., s_m)^\top \in \mathbb{R}^m$ is given by

$$[\text{Prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{s})]_i := \begin{cases} 0, & 0 < s_i \le \sqrt{2\gamma C}, \\ s_i, & s_i > \sqrt{2\gamma C} \text{ or } s_i \le 0. \end{cases} \tag{S5}$$

Proof. It follows from (S4) that $[\text{Prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{s})]_i = \text{Prox}_{\gamma C\|(\cdot)_+\|_0}(s_i)$, where

$$\text{Prox}_{\gamma C\|(\cdot)_+\|_0}(s_i) = \arg\min_{v\in\mathbb{R}} C\|v_+\|_0 + \frac{1}{2\gamma}(v-s_i)^2.$$

Using (S3) completes the proof. □

S.3 Proof of Theorem 3.2

(i) By assumption that B defined by (13) is full column rank, $B^+ = (B^\top B)^{-1}B^\top$ exists. So the constraint in (9) can be written as $(\mathbf{w}; b) = B^+(\mathbf{1} - \mathbf{u})$ due to $\mathbf{1} - \mathbf{u} = A\mathbf{w} + b\mathbf{y} = B(\mathbf{w}; b)$. Because of this, (9) becomes

$$\min_{\mathbf{u}\in\mathbb{R}^m} \ \frac{1}{2}\|H(\mathbf{u}-\mathbf{1})\|^2 + C\|\mathbf{u}_+\|_0, \tag{S6}$$

where H is defined in (13). Suppose $\mathbf{u}^*$ is a globally optimal one to (9), which is also a globally optimal solution to (9) since (9) is equivalent to (9) when B is full column rank. To show the conclusion (namely to show (11)). We first prove a) the globally optimal $\mathbf{u}^*$ satisfying

$$\mathbf{u}^* \ = \text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma H^\top H(\mathbf{u}^* - \mathbf{1})), \tag{S7}$$

and then show b) (S7) implying (11). For simplicity, let

$$\begin{aligned} g(\mathbf{u}) &:= \|H(\mathbf{u}-\mathbf{1})\|^2/2, \\ \boldsymbol{\lambda}^* &:= H^\top H(\mathbf{u}^* - \mathbf{1}). \end{aligned} \tag{S8}$$

To see a), it is sufficient to prove $\mathbf{u}^* = \mathbf{z}$, where $\mathbf{z} := \text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*)$, before which, we need three facts. The first one is from the definition (S4) of the $L_{0/1}$ proximal operator that

$$\begin{aligned} & C\|\mathbf{z}_+\|_0 + \frac{1}{2\gamma}\|\mathbf{z} - (\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*)\|^2 \\ \le\ & C\|\mathbf{u}^*{}_+\|_0 + \frac{1}{2\gamma}\|\mathbf{u}^* - (\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*)\|^2 \\ =\ & C\|\mathbf{u}^*{}_+\|_0 + \frac{\gamma}{2}\|\boldsymbol{\lambda}^*\|^2. \end{aligned} \tag{S9}$$

The second fact follows from g being quadratic that

$$g(\mathbf{z}) - g(\mathbf{u}^*) \le \langle \boldsymbol{\lambda}^*, \mathbf{z} - \mathbf{u}^* \rangle + \frac{\lambda_H}{2}\|\mathbf{z} - \mathbf{u}^*\|^2. \tag{S10}$$

The last fact from the global optimality of $\mathbf{u}^*$ to (9) is

$$C\|\mathbf{u}^*{}_+\|_0 + g(\mathbf{u}^*) \le C\|\mathbf{z}_+\|_0 + g(\mathbf{z}). \tag{S11}$$

Three facts lead to the following chain of inequalities,

$$\begin{aligned} 0 \overset{\text{(S11)}}{\le}\ & C\|\mathbf{z}_+\|_0 + g(\mathbf{z}) - (C\|\mathbf{u}^*{}_+\|_0 + g(\mathbf{u}^*)) \\ \overset{\text{(S10)}}{\le}\ & C\|\mathbf{z}_+\|_0 + \langle \boldsymbol{\lambda}^*, \mathbf{z} - \mathbf{u}^* \rangle + \frac{\lambda_H}{2}\|\mathbf{z} - \mathbf{u}^*\|^2 - C\|\mathbf{u}^*{}_+\|_0 \\ =\ & C\|\mathbf{z}_+\|_0 + \langle \boldsymbol{\lambda}^*, \mathbf{z} - \mathbf{u}^* \rangle + \frac{1}{2\gamma}\|\mathbf{z} - \mathbf{u}^*\|^2 - C\|\mathbf{u}^*{}_+\|_0 \\ -\ & \frac{1}{2\gamma}\|\mathbf{z} - \mathbf{u}^*\|^2 + \frac{\lambda_H}{2}\|\mathbf{z} - \mathbf{u}^*\|^2 \\ =\ & C\|\mathbf{z}_+\|_0 + \frac{1}{2\gamma}\|\mathbf{z} - (\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*)\|^2 - \frac{\gamma}{2}\|\boldsymbol{\lambda}^*\|^2 \\ -\ & C\|\mathbf{u}^*{}_+\|_0 + \frac{\lambda_H - 1/\gamma}{2}\|\mathbf{z} - \mathbf{u}^*\|^2 \\ \overset{\text{(S9)}}{\le}\ & \frac{\lambda_H - 1/\gamma}{2}\|\mathbf{z} - \mathbf{u}^*\|^2, \end{aligned} \tag{S12}$$

which indicates $\|\mathbf{z} - \mathbf{u}^*\|^2 \le 0$ due to $0 < \gamma < 1/\lambda_H$. Hence, we have $\mathbf{z} = \mathbf{u}^*$, as claimed.

Next we show b). It follows from (S8) that

$$-\boldsymbol{\lambda}^* = H^\top H(\mathbf{1} - \mathbf{u}^*) = H^\top E B^+(\mathbf{1} - \mathbf{u}^*) = H^\top E \begin{bmatrix} \mathbf{w}^* \\ b^* \end{bmatrix},$$

where $E := \begin{bmatrix} I_{n\times n} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix}$, which suffices to

$$-B^\top\boldsymbol{\lambda}^* = B^\top H^\top E \begin{bmatrix} \mathbf{w}^* \\ b^* \end{bmatrix} = B^\top (B^+)^\top E^\top E \begin{bmatrix} \mathbf{w}^* \\ b^* \end{bmatrix} = \begin{bmatrix} \mathbf{w}^* \\ 0 \end{bmatrix},$$

where we used two facts $B^\top(B^+)^\top = B^\top B(B^\top B)^{-1} = I$ and $E^\top E = E$. By the definition (13) that $B = [A\ \mathbf{y}]$, the above equation yields

$$\begin{cases} \mathbf{w}^* + A^\top\boldsymbol{\lambda}^* &= \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\lambda}^* \rangle &= 0. \end{cases}$$

Finally, the above conditions, the feasibility of $(\mathbf{w}^*; b^*; \mathbf{u}^*)$ and (S7) lead to (11), which completes the proof of result (i).

(ii) Suppose $\boldsymbol{\phi}^* := (\mathbf{w}^*; b^*; \mathbf{u}^*)$ is a P-stationary point of (9) with $\gamma > 0$, then there is a $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that $(\boldsymbol{\phi}^*; \boldsymbol{\lambda}^*)$ satisfies (11), i.e.,

$$\begin{cases} \mathbf{w}^* + A^\top\boldsymbol{\lambda}^* &= \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\lambda}^* \rangle &= \mathbf{0}, \\ \mathbf{u}^* + A\mathbf{w}^* + b^*\mathbf{y} &= \mathbf{1}, \\ \text{prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{u}^* - \gamma\boldsymbol{\lambda}^*) &= \mathbf{u}^*, \end{cases} \tag{S13}$$

where

$$[\mathrm{Prox}_{\gamma C\|(\cdot)_+\|_0}(\mathbf{z}^*)]_i = \begin{cases} 0, & 0 < z_i^* \le \sqrt{2\gamma C}, \\ z_i^*, & z_i^* > \sqrt{2\gamma C} \text{ or } z_i^* \le 0, \end{cases} \tag{S14}$$

and $\mathbf{z}^* = \mathbf{u}^* - \gamma\boldsymbol{\lambda}^*$. Let Θ be the feasible region of (9), namely,

$$\Theta := \{\boldsymbol{\phi} := (\mathbf{w}; b; \mathbf{u}) : \mathbf{u} + A\mathbf{w} + b\mathbf{y} = \mathbf{1}\}. \tag{S15}$$

Furthermore, the function $||\mathbf{u}_+||_0$ is lower semi-continuous at $\boldsymbol{\phi}^* \in \Theta$, then by [1, Proposition 4.3], there is a neighborhood $U(\boldsymbol{\phi}^*, \delta_1)$ of $\boldsymbol{\phi}^* \in \Theta$ with $\delta_1 > 0$ such that

$$||\mathbf{u}_+||_0 > ||\mathbf{u}_+^*||_0 - \frac{1}{2}, \forall \boldsymbol{\phi} \in \Theta \cap U(\boldsymbol{\phi}^*, \delta_1). \tag{S16}$$

In addition, since $\|\mathbf{w}\|^2$ is locally lipschitz continuous in $\mathbb{R}^n$, there exists a neighborhood $U(\boldsymbol{\phi}^*, \delta_2)$ of $\boldsymbol{\phi}^* \in \Theta$ with $\delta_2 > 0$ such that

$$|\|\mathbf{w}\|^2 - \|\mathbf{w}^*\|^2| \le 2C, \ \ \forall \boldsymbol{\phi} \in \Theta \cap U(\boldsymbol{\phi}^*, \delta_2). \tag{S17}$$

Denote $\delta := \min\{\delta_1, \delta_2\}$. Now we show that $\boldsymbol{\phi}^*$ is a local minimizer of (9). Namely, there exists a neighborhood $U(\boldsymbol{\phi}^*, \delta)$ of $\boldsymbol{\phi}^* \in \Theta$ with $\delta > 0$ such that

$$\begin{array}{rcll} \frac{1}{2}\|\mathbf{w}^*\|^2 + C\|\mathbf{u}_+^*\|_0 & \le & \frac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0, & \text{(S18)} \\ & & \forall \boldsymbol{\phi} \in \Theta \cap U(\boldsymbol{\phi}^*, \delta). & \end{array}$$

For this purpose, let $\Gamma_* := \{i : u_i^* = 0\}$ and $\overline{\Gamma}_* := \mathbb{N}_m \backslash \Gamma_*$. It follows from the fourth equation of (S13) and (S14) that we obtain

$$\begin{array}{c} -\sqrt{2C/\gamma} \le \lambda_i^* \le 0, \ u_i^* = 0, \ \ \forall i \in \Gamma_*, \\ \lambda_i^* = 0, \ u_i^* \ne 0, \ \ \forall i \in \overline{\Gamma}_*. \end{array} \tag{S19}$$

Based on these, we consider a local region Θ_1 of Θ as

$$\Theta_1 := \Theta \cap \{\boldsymbol{\phi} : u_i \le 0, \ \ i \in \Gamma_*\}. \tag{S20}$$

We split the proof of the (S18) into the following two cases:

Case (i): $\boldsymbol{\phi} \in \Theta_1 \subseteq \Theta$ and $\boldsymbol{\phi} \in U(\boldsymbol{\phi}^*, \delta)$. It is easy to see that $\boldsymbol{\phi}^* \in \Theta_1$ by (S13). Then for any $\boldsymbol{\phi} \in \Theta_1$, we have two facts

$$u_i \le 0, \ i \in \Gamma_* \tag{S21}$$

and $\mathbf{u} + A\mathbf{w} + b\mathbf{y} = \mathbf{1}$, which and (S13) suffice to

$$-A(\mathbf{w} - \mathbf{w}^*) = \mathbf{u} - \mathbf{u}^* + (b - b^*)\mathbf{y}. \tag{S22}$$

The following chain of inequalities hold for any $\boldsymbol{\phi} \in \Theta_1$,

$$\begin{array}{rcl} & \|\mathbf{w}\|^2 - \|\mathbf{w}^*\|^2 & \\ = & \|\mathbf{w} - \mathbf{w}^* + \mathbf{w}^*\|^2 - \|\mathbf{w}^*\|^2 & \\ = & 2\langle \mathbf{w} - \mathbf{w}^*, \mathbf{w}^* \rangle + \|\mathbf{w} - \mathbf{w}^*\|^2 & \\ \ge & 2\langle \mathbf{w} - \mathbf{w}^*, \mathbf{w}^* \rangle & \\ \overset{\text{(S13)}}{=} & -2\langle \mathbf{w} - \mathbf{w}^*, A^\top \boldsymbol{\lambda}^* \rangle & \\ = & -2\langle A(\mathbf{w} - \mathbf{w}^*), \boldsymbol{\lambda}^* \rangle & \\ \overset{\text{(S22)}}{=} & 2\langle \mathbf{u} - \mathbf{u}^*, \boldsymbol{\lambda}^* \rangle + 2(b - b^*)\langle \mathbf{y}, \boldsymbol{\lambda}^* \rangle & \\ \overset{\text{(S13)}}{=} & 2\langle \mathbf{u} - \mathbf{u}^*, \boldsymbol{\lambda}^* \rangle & \\ = & 2\langle \mathbf{u}_{\Gamma_*} - \mathbf{u}_{\Gamma_*}^*, \boldsymbol{\lambda}_{\Gamma_*}^* \rangle + 2\langle \mathbf{u}_{\overline{\Gamma}_*} - \mathbf{u}_{\overline{\Gamma}_*}^*, \boldsymbol{\lambda}_{\overline{\Gamma}_*}^* \rangle & \\ \overset{\text{(S19)}}{=} & 2\langle \mathbf{u}_{\Gamma_*}, \boldsymbol{\lambda}_{\Gamma_*}^* \rangle & \\ \overset{\text{(S19),(S21)}}{\ge} & 0. & \text{(S23)} \end{array}$$

Since $||u_+||_0$ can only take values from $\{0, 1, \cdots, m\}$, this together with (S16) allows us to conclude that

$$||\mathbf{u}_+||_0 \ge ||\mathbf{u}_+^*||_0, \forall \boldsymbol{\phi} \in \Theta \cap U(\boldsymbol{\phi}^*, \delta_1). \tag{S24}$$

Therefore, for any $\boldsymbol{\phi} \in \Theta_1 \cap U(\boldsymbol{\phi}^*, \delta) \subseteq \Theta_1 \cap U(\boldsymbol{\phi}^*, \delta_1)$, then (S23) and (S24) lead to

$$\frac{1}{2}\|\mathbf{w}^*\|^2 + C\|\mathbf{u}_+^*\|_0 \le \frac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0. \tag{S25}$$

Case (ii): $\boldsymbol{\phi} \in (\Theta \setminus \Theta_1)$ and $\boldsymbol{\phi} \in U(\boldsymbol{\phi}^*, \delta)$. It follows from that $\boldsymbol{\phi} \in (\Theta \setminus \Theta_1)$ that there exists $i_0 \in \Gamma_*$ with $u_{i_0}^* = 0$ but $u_{i_0} > 0$, which implies $\|(u_{i_0}^*)_+\|_0 = 0$ but $\|(u_{i_0})_+\|_0 = 1$. By $\boldsymbol{\phi} \in U(\boldsymbol{\phi}^*, \delta)$ and (S24), we have

$$\|\mathbf{u}_+\|_0 \ge \|\mathbf{u}_+^*\|_0 + 1. \tag{S26}$$

This together with (S17) obtains that for any $\boldsymbol{\phi} \in (\Theta \setminus \Theta_1) \cap U(\boldsymbol{\phi}^*, \delta)$,

$$\begin{array}{rcl} \frac{1}{2}\|\mathbf{w}^*\|^2 + C\|\mathbf{u}_+^*\|_0 & \le & \frac{1}{2}\|\mathbf{w}^*\|^2 + C\|\mathbf{u}_+\|_0 - C \\ & \le & \frac{1}{2}\|\mathbf{w}\|^2 + C\|\mathbf{u}_+\|_0. \end{array} \tag{S27}$$

Summarizing (S25) and (S27), we obtain that $\boldsymbol{\phi}^*$ is a local minimizer of (9) in a local region $\Theta \cap U(\boldsymbol{\phi}^*, \delta)$, which completes the proof. □

S.4 Proof of Theorem 4.1

Since $T_k \subseteq \mathbb{N}_m$ has finite many elements, for sufficient large k, there is a subset $J \subseteq \{1, 2, 3, \cdots\}$ such that

$$T_j \equiv: T, \ \ \forall\, j \in J. \tag{S28}$$

For notational simplicity, denote the sequence $\boldsymbol{\Psi}^k := (\mathbf{w}^k, b^k, \mathbf{u}^k, \boldsymbol{\lambda}^k)$ and its limit point $\boldsymbol{\Psi}^* := (\mathbf{w}^*, b^*, \mathbf{u}^*, \boldsymbol{\lambda}^*)$, namely $\{\boldsymbol{\Psi}^k\} \to \boldsymbol{\Psi}^*$. This also indicates $\{\boldsymbol{\Psi}^j\}_{j\in J} \to \boldsymbol{\Psi}^*$ and $\{\boldsymbol{\Psi}^{j+1}\}_{j\in J} \to \boldsymbol{\Psi}^*$. Taking the limit along with J of (32), namely, $k \in J, k \to \infty$, we have

$$\left\{ \begin{array}{rcl} \boldsymbol{\lambda}_T^* & = & \boldsymbol{\lambda}_T^* + \eta\sigma\boldsymbol{\varpi}_T^*, \\ \boldsymbol{\lambda}_{\overline{T}}^* & = & \mathbf{0}, \end{array} \right. \tag{S29}$$

which derives $\boldsymbol{\varpi}_T^* = \mathbf{0}$. Taking the limit along with J of (25) and $\mathbf{z}^k$ respectively yields

$$\begin{array}{rcll} \mathbf{z}^* & = & \mathbf{1} - A\mathbf{w}^* - b^*\mathbf{y} - \boldsymbol{\lambda}^*/\sigma & \\ & = & \mathbf{1} - A\mathbf{w}^* - b^*\mathbf{y} - \mathbf{u}^* + \mathbf{u}^* - \boldsymbol{\lambda}^*/\sigma & \\ & = & -\boldsymbol{\varpi}^* + \mathbf{u}^* - \boldsymbol{\lambda}^*/\sigma & \text{(S30)} \end{array}$$

and thus

$$\begin{array}{rcll} \mathbf{u}_T^* & = & \mathbf{0}, & \text{(S31)} \\ \mathbf{u}_{\overline{T}}^* & = & \mathbf{z}_{\overline{T}}^* & \text{(S32)} \\ & \overset{(S30)}{=} & -\boldsymbol{\varpi}_{\overline{T}}^* + \mathbf{u}_{\overline{T}}^* - \boldsymbol{\lambda}_{\overline{T}}^*/\sigma & \\ & \overset{(S29)}{=} & -\boldsymbol{\varpi}_{\overline{T}}^* + \mathbf{u}_{\overline{T}}^*. & \end{array}$$

This proves $\boldsymbol{\varpi}_{\overline{T}}^* = \mathbf{0}$ and hence $\boldsymbol{\varpi}^* = \mathbf{0}$. Again by (S30), we obtain $\mathbf{z}^* = \mathbf{u}^* - \boldsymbol{\lambda}^*/\sigma$, which together with (S31), (S32) and the Definition S.2 of the $L_{0/1}$ proximal operator indicates

$$\mathbf{u}^* = \mathrm{Prox}_{\frac{C}{\sigma}\|(\cdot)_+\|_0}(\mathbf{z}^*) = \mathrm{Prox}_{\frac{C}{\sigma}\|(\cdot)_+\|_0}(\mathbf{u}^* - \boldsymbol{\lambda}^*/\sigma). \tag{S33}$$

Now taking the limit along with J of (28) results in

$$\begin{array}{rcl}(I+\sigma A_T^\top A_T)\mathbf{w}^* &=& \sigma A_T^\top \mathbf{v}_T^* \\ &=& -\sigma A_T^\top(\mathbf{u}_T^* + b^*\mathbf{y}_T - \mathbf{1} + \boldsymbol{\lambda}_T^*/\sigma) \\ &=& -\sigma A_T^\top(\boldsymbol{\varpi}_T^* - A_T\mathbf{w}^* + \boldsymbol{\lambda}_T^*/\sigma) \\ &=& -\sigma A_T^\top(-A_T\mathbf{w}^* + \boldsymbol{\lambda}_T^*/\sigma),\end{array}$$

where $\mathbf{v}^* = -(\mathbf{u}^* + b^*\mathbf{y} - \mathbf{1} + \boldsymbol{\lambda}^*/\sigma)$ and the last two equations hold due to $\boldsymbol{\varpi}^* = \mathbf{u}^* + A\mathbf{w}^* + b^*\mathbf{y} - \mathbf{1} = \mathbf{0}$. The last equation suffices to that

$$\mathbf{w}^* \;=\; -A_T^\top \boldsymbol{\lambda}_T^* \overset{(S29)}{=} -A^\top \boldsymbol{\lambda}^*.$$

Finally taking the limit along with J of (31) leads to

$$\begin{array}{rcl}b^* = \langle \mathbf{y}, \mathbf{r}^*\rangle/m &=& -\langle \mathbf{y}, A\mathbf{w}^* - \mathbf{1} + \mathbf{u}^* + \boldsymbol{\lambda}^*/\sigma\rangle/m \\ &=& -\langle \mathbf{y}, \boldsymbol{\varpi}^* - b^*\mathbf{y} + \boldsymbol{\lambda}^*/\sigma\rangle/m \\ &=& -\langle \mathbf{y}, -b^*\mathbf{y} + \boldsymbol{\lambda}^*/\sigma\rangle/m \\ &=& b^* - \langle \mathbf{y}, \boldsymbol{\lambda}^*\rangle/(m\sigma),\end{array}$$

which contributes to $\langle \mathbf{y}, \boldsymbol{\lambda}^*\rangle = 0$. Overall, we have

$$\left\{\begin{array}{rcl}\mathbf{w}^* + A^\top\boldsymbol{\lambda}^* &=& \mathbf{0}, \\ \langle \mathbf{y}, \boldsymbol{\lambda}^*\rangle &=& \mathbf{0}, \\ \mathbf{u}^* + A\mathbf{w}^* + b^*\mathbf{y} &=& \mathbf{1}, \\ \mathrm{prox}_{\frac{C}{\sigma}\|(\cdot)_+\|_0}(\mathbf{u}^* - \boldsymbol{\lambda}^*/\sigma) &=& \mathbf{u}^*.\end{array}\right.$$

Namely, $(\mathbf{w}^*; b^*; \mathbf{u}^*)$ is a P-stationary point of problem (9) where $\gamma = 1/\sigma$. Then by Theorem 3.2 (ii), it is a locally optimal solution to the problem (9), which completes the proof. □

REFERENCES

[1] B. Mordukhovich and N. Nam, "An easy path to convex analysis and applications", *Morgan and Claypool Pubulishers,* 2014.

[2] L. Guan, L. B. Qiao, D. S. Li, T. Sun, K. S. Ge, and X. C. Lu, "An efficient ADMM-based algorithm to nonconvex penalized support vector machines", *in Proc. Int. Conf. Data Mining Workshops,* 2018, 1209-1216.

[3] Q. M. Yao, J. T. Kwok, F. Gao, W. Chen, and T. Y. Liu, "Efficient inexact proximal gradient algorithm for nonconvex problems", *arXiv preprint arXiv:1612.09069,* 2016.

[4] B. Gu, Z. Y Huo, and H. Huang, "Inexact proximal gradient methods for non-convex and non-smooth optimization", *in Proc. AAAI Conf. Artif. Intell.,* 2018, pp. 3093-3100.

[5] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations", *J. Fourier Anal.Appli.,* vol. 14, no. 5-6, pp. 629-654, 2008.

[6] T. Blumensath and M. E. Davies, "Iterative hard thresholding for compressed sensing", *Appl. Comput. Harmonic Anal.,* vol. 27, no. 3, pp. 265-274, 2009.

[7] Z. S. Lu and Y. Zhang, "Sparse approximation via penalty decomposition methods", *SIAM J. Optim.,* vol. 23, no. 4, pp. 2448-2478, 2013.

[8] Z. S. Lu, "Iterative reweighted minimization methods for l_p-regularized unconstrained nonlinear programming", *Math. Program.,* vol. 147, no.1-2, pp. 277-307, 2014.

[9] A. Beck and N. Hallak, "Proximal mapping for symmetric penalty and sparsity", *SIAM J. Optim.,* vol. 28, no. 1, pp. 496-527, 2018.

[10] H. Zhang, L. L. Pan, and N. H. Xiu, "Optimality conditions for locally Lipschitz optimization with l_0-regularization", *Optim. Lett.,* DOI: 10.1007/s11590-020-01579-y, 2020.